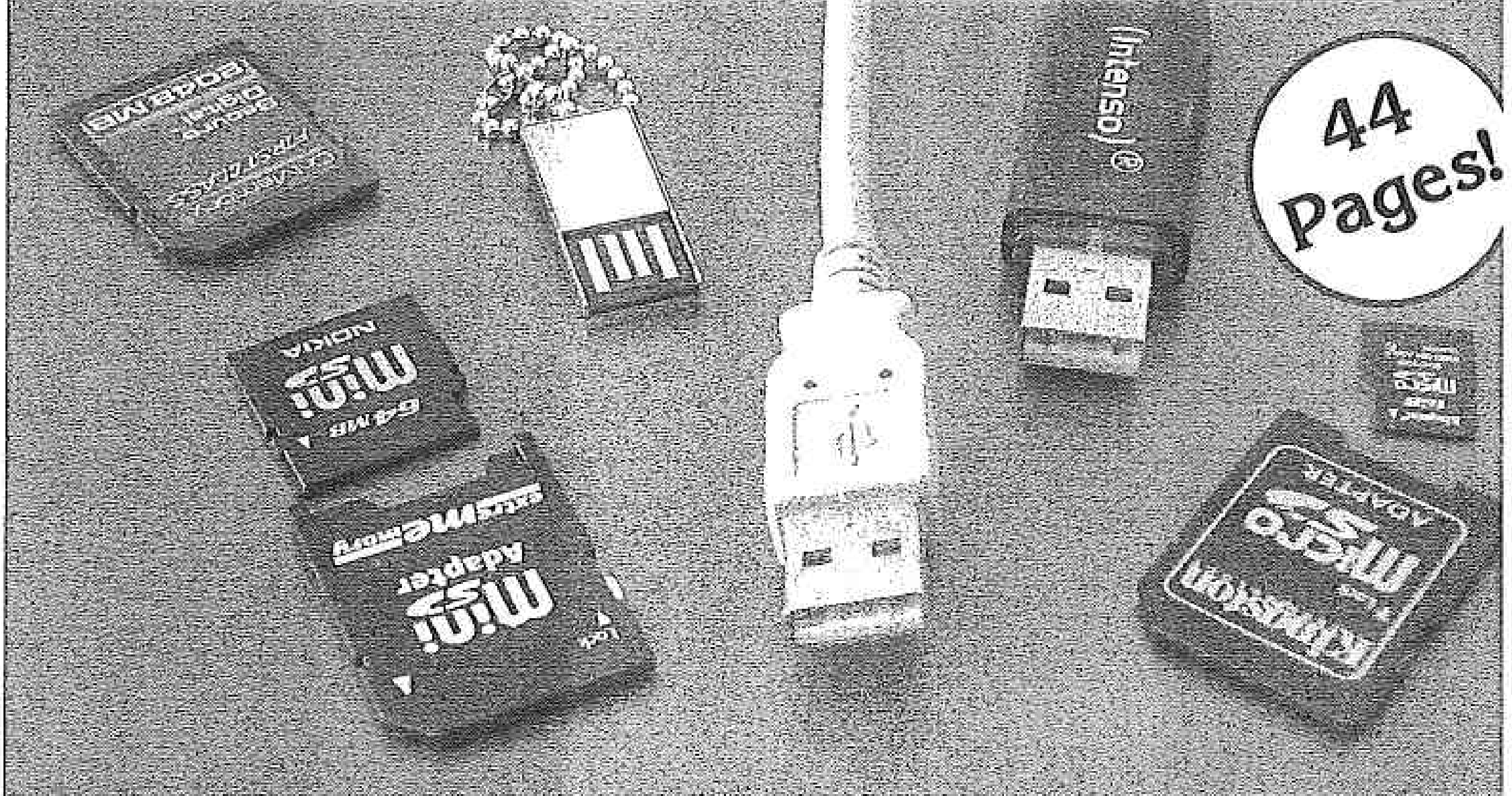
QL Today

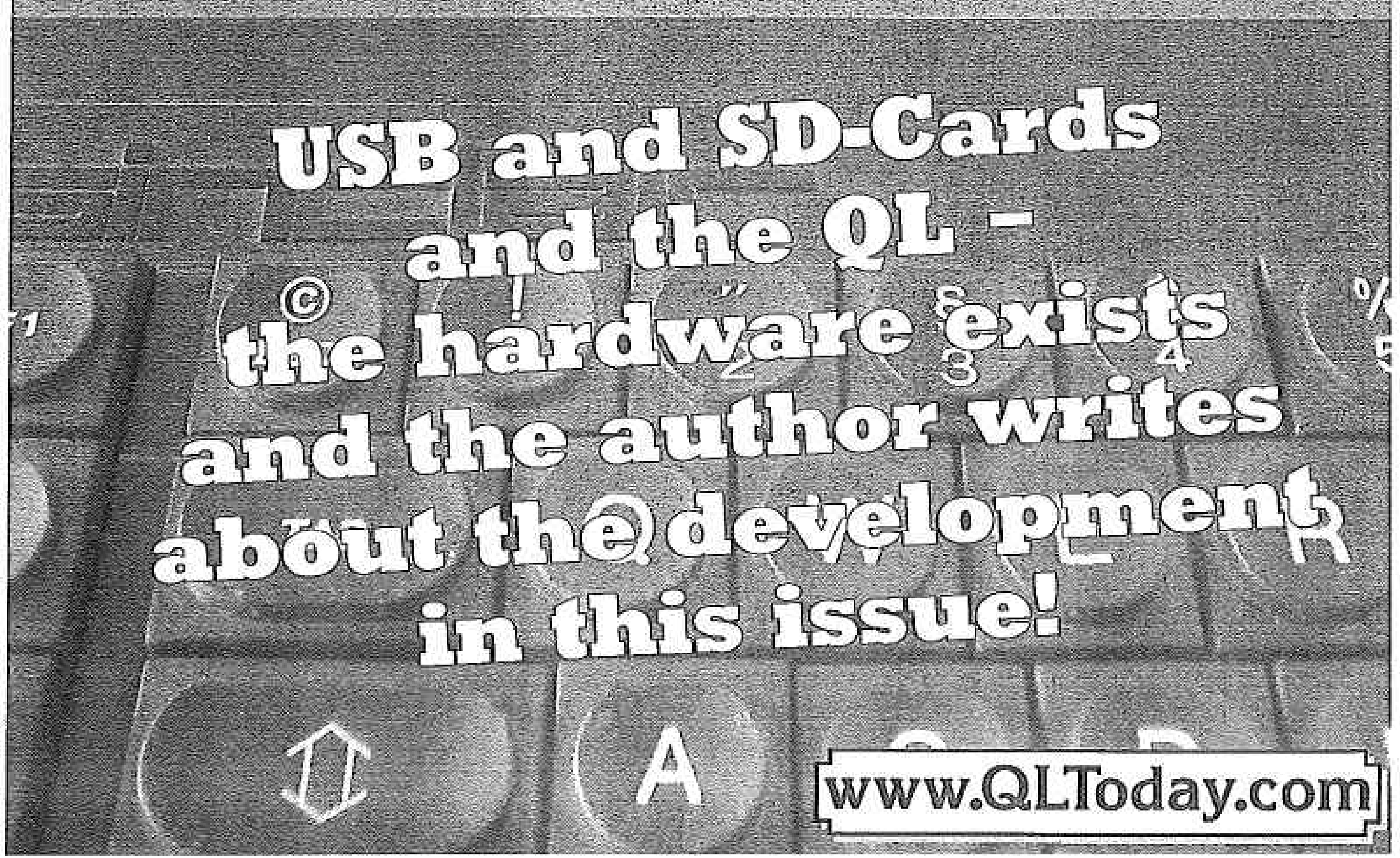
Volume 15
Issue 3
Feb. - April
2011

ISSN 1432-5454

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...

44 Pages!





USB and SD-Cards and the QL - the hardware exists and the author writes about the development in this issue!

www.QLToday.com

Contents

The deadline for the next issue is the 15th of May 2011

Advertisers

in alphabetical order

QL *Today*

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software	Tel. +49 203 502011
Kaiser-Wilhelm-Str. 302	Fax +49 203 502012
47169 Duisburg	email: smsq@j-m-s.com
Germany	email: QLToday@j-m-s.com

Editor:

Geoff Wicks	Tel. +44 1332 271366
Flat 5b	email: gtwicks@btinternet.com
Wordsworth Avenue	email: QLToday@j-m-s.com
Derby DE24 9HQ	
United Kingdom	

Co-Editor & UK Office:

Bruce Nicholls	Tel +44 20 71930539
38 Derham Gardens	Fax +44 870 0568755
Upminster	email: qltoday@q-v-d.demon.co.uk
Essex RM14 3HA	email: QLToday@j-m-s.com
United Kingdom	

QL *Today* is published four times a year, our volume begins on beginning of June. Please contact the German or English office for current subscription rates or visit our homepage www.QLTODAY.com.

We welcome your comments, suggestions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5' disk (DD or HD) or sent via Email. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

If you need more information about the UNZIP program which is used by our BOOT program to unpack the files, we suggest that you visit Dilwyn Jones' web site where you find more information about lots of interesting QDOS software and INFOZIP at http://www.dilwyn.uk6.net/arch/index.html

Editorial

by Geoff Wicks

The QL has started the new year bursting with life. The new forum, started at the end of last year, has firmly established itself, and the QL-users email list, that a year ago appeared moribund, has suddenly blossomed. On occasions there are now over 20 postings per day.

In the last year there has been a revival of interest in hardware especially native hardware. Several small scale projects for native hardware are being actively discussed. In this issue is an article over a USB project at an advanced stage, and news of a possible cardreader to fit in the microdrive slot. Too late for inclusion in our main news section were proposals for a new backplane and a battery holder for Gold Cards using the popular 2032 battery.

On the negative side we report on technical problems with the Quanta website. At QL Today we have known of these problems for some months, but decided against publication. Partly because members of the Quanta Committee were unwilling to comment on the record - we had to piece our story together from various sources. And partly because we knew the committee were holding a meeting to solve the problems. Almost three months after that meeting there is still no sign of a solution, and the time has come to break our silence.

A worrying feature of the website failure is the unwillingness of Quanta's officers to share the problem with the members. It echoes the situation three years ago when Quanta failed to inform the members about serious problems with the Quanta Magazine causing some to think Quanta had closed down. If Quanta's officers are unwilling to share bad news they are creating a distance between themselves and the members. Can they then complain about the passivity of the members?

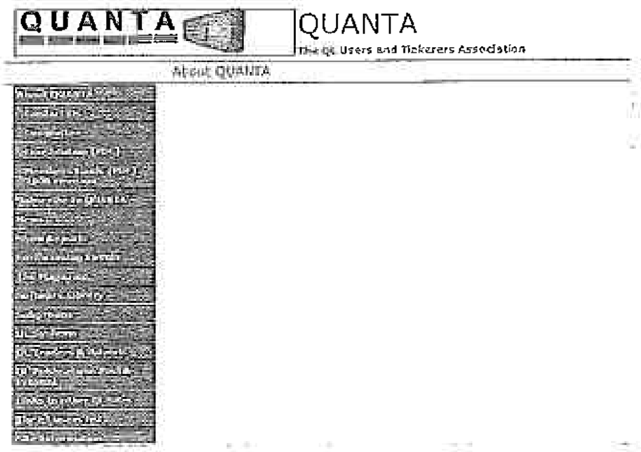
If Quanta had been open about the technical problems with the website, they would have found sympathy and offers of help. At least that was our experience at QL Today when we shared our problems over producing an electronic issue. There was a lengthy discussion on the QL users email list and several people offered practical help. We are no nearer solving the main problem, but at least we are now able to produce an electronic edition for archiving purposes.

We also have the sad news is that we are shortly to lose one of our regular writers. Steve Poole has had a prolific output not only in QL Today but also in the Quanta Magazine. His speciality was short SuperBasic programs and that type of article is not so common these days. Would someone like to take over the task? Although we still have a few of Steve's programs on file, we are printing his farewell message in this issue in co-ordination with the Quanta Magazine. We are grateful to Steve, who will be sorely missed, and his faithful, but unseen, beta tester Bruno Coativy, for their contributions to QL Today.

Nevertheless the positives still outweigh the negatives. In the last few months a handful of former QL-ers have returned to the scene and they have brought with them a lot of enthusiasm and fresh thinking. Those of us who stayed with the QL throughout should be humble enough to realise we have much to learn from them. They could help us overcome the malaise in the UK scene in general and Quanta in particular.

QUANTA'S Web Collapse

The Quanta website has been hit by serious technical problems that have prevented it from being updated for over 6 months. QL Today understands the new Content Management System (CMS), introduced at the beginning of last year, has suffered a massive failure. The cause of this is proving elusive. Individual Quanta committee members are reluctant to speak on the record about the problems, but a number of suggestions for the failure are being made. These include Quanta's aspirations for the site being too ambitious; the lack of technical support from a free host; and the difficulties of integrating a Linux based CMS into a Windows environment.
The problems of the website are preventing the implementation of the restricted members' area. QL Today understands several committee members completed their work to implement the members' area several months ago. These include the passwords members will need and the total contents of the Quanta library. Another victim is the news pages, which have not been updated since July 2010. Quanta has also been unable to publicise its forthcoming AGM on its website, which shows the next event as being last year's Austrian show.

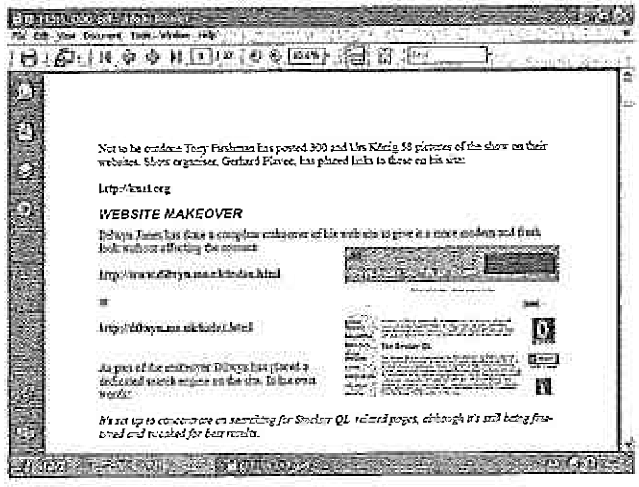


The Quanta website has frequently had display problems on Internet Explorer, but not on other browsers. Internet Explorer correctly displays the menu items on the left hand side, but the rest of the screen appears blank until the user scrolls down to below the menu.
In what is reminiscent of a magazine crisis three years ago, Quanta's officers have failed to inform the members of the problems. After an emergency committee meeting to resolve the problems at the beginning of December, Secretary Alison Southern wrote somewhat cryptically in the Quanta Magazine,
"Talking about the website, I think we are being close to being ready to launch on this, more information will be found in this issue"
There was no further information in the rest of the magazine and over two months later no signs of the problems being resolved.
Quanta has had a website for over ten years, but has never succeeded in keeping it up to date. In the past the problem has been uncertainty over who had responsibility for the editorial content of the site and the new CMS was designed to overcome this problem. This time the editorial work is well advanced, but the CMS problems are preventing it being placed online.

Electronic QL TODAY Progress

Norman Dunbar has continued his work to investigate the feasibility of a electronic version of QL Today using a sample of files from Volume 5 issue 1. Using Open Office he has produced a PDF version with a file size of 716KB. However this was using a simpler layout than QL Today and without the detailed page by page design necessary in the published magazine. Norman's design had 37 pages compared with the 30 pages the articles took up in QL Today. To keep the weight of the magazine within the lower postage rates QL Today has to squeeze more information onto the individual pages and this would lead to some increase in file size.
QL Today asked Norman if he could estimate how the file size would change if a two column design had been used instead. He immediately transformed his design to a two column layout producing a file size of 756KB.
Norman reports that Open Office is easier to use for a magazine layout than Microsoft Word because it is simpler to anchor illustrations. However both QL magazines prefer to use dedicated desk top publishing programs. The Quanta Magazine is produced in Serif PagePlus and QL Today in Calamus, which is a PC program emulating an Atari program.
Norman's work would indicate that an electronic QL Today could have a small file size, but does not solve QL Today's immediate problem that Calamus can only produce large bit map files.
During the course of his work Norman also converted the files to .epub format for use on an ereader device. He reported a possible clash bet-

ween the ideal design requirements for a paper magazine and an electronic one. There is a relationship between ease of reading and a combination of font size and column width. For this reason most A4 magazines use a two or more column format. However, this type of format is more difficult to read on an ereader.



Not to be outdone Tony Firshman has posted 300 and Urs König 58 pictures of the show on their websites. Show organiser, Gerhard Plavec, has placed links to these on his site:

http://[illegible].org

WEBSITE MAKEOVER

Dilwyn Jones has done a complete makeover of his web site to give it a more modern and fresh look without affecting the content.

http://www.dilwyn.me.uk/index.html

or

http://dilwyn.me.uk/index.html

As part of the makeover Dilwyn has placed a dedicated search engine on the site. In his own words:

It's set up to concentrate on searching for Sinclair QL related pages, although it's still being fine-tuned and tweaked for best results.

Several people have suggested that a magazine produced in pdf format would be difficult to adapt for loading into ereaders. As an alternative Word or web formats were suggested.

QL Today has plans to produce an electronic version of the present volume for archiving purposes.

QL Forum

As briefly mentioned in a late news item in the last issue of QL Today Peter Scott has launched a QL Forum at:

www.qlforum.co.uk

The forum is divided into 4 sections:

FORUM: This gives the site announcements and rules.

GENERAL: This is divided into several subsections - Help for new users; QL chat; Hardware; Software; Emulators; and QL compatible computers.

MARKET PLACE: A buying and selling area.

FORUM: This is for off topic items.

Anyone can view the forum, but people who wish to contribute must first register as members. At the beginning of February there had been 367 posts on 84 different topics. 44 people had registered as members.

Hardware Plans

Recently there have been several hardware projects discussed on the QL-Users email group. **Adrian Ives** wrote:

"I have no idea if anyone is remotely interested in this project to attach USB devices to a QL using a small card called a USBWiz. This device presents a serial interface and accepts AT style commands to communicate with many classes of USB device. I started working on this last year, but was delayed by some family problems and a move to another part of the country. My prototype hardware is a little black block that connects via a serial lead to a QL or Hermes serial port. The box has an SD card slot and two USB ports."

Many people were interested in this project and elsewhere in this issue Adrian describes the project in some detail. Adrian does warn there is still a long way to go before he has a working unit.

Peter Graf raised the possibility of an SD/MMC card hard disk for the QL and asked which style people would prefer:

"I can not promise to really make a piece of hardware available, but it would be nice to know, just in case...

A) External interface, plugs into parallel port of Super Gold Card
Pro: - Interface also works on Q40 and Q60
- QL Case doesn't need to be opened
- Easy reconnect from one machine to another
- Hot-plugging might work
- ROM-Port remains usable
Con: - Slow data transfer through parallel port handshake lines

B) External interface, plugs into QL ROM port
Pro: - QL Case doesn't need to be opened
- Faster data transfer
- Onboard Driver ROM
- Works on QL without Gold Card/ Super Gold Card
Con: - ROM-Port occupied
- Complex hardware

C) Internal interface, plugs into CPU socket
Pro: - Fastest data transfer
- ROM-Port remains usable
Con: - QL Case needs to be opened
- Only Gold Card/Super Gold Card machines

D) Internal interface, replacing a microdrive(!). Can easily be bolted inside the case, after a microdrive was removed Plugs into CPU socket or maybe other place
Pro: - SD/MMC-card can be plugged in like a cartridge

- Looks cool
- Very "QL-style"
- ROM-Port remains usable
Con: - QL Case needs to be opened"

The voting was overwhelmingly in favour of using a microdrive socket, with the ROM slot and Gold Cold parallel ports coming a poor second and third respectively.

Dilwyn Jones

Dilwyn has several news items:

WORDSPOT

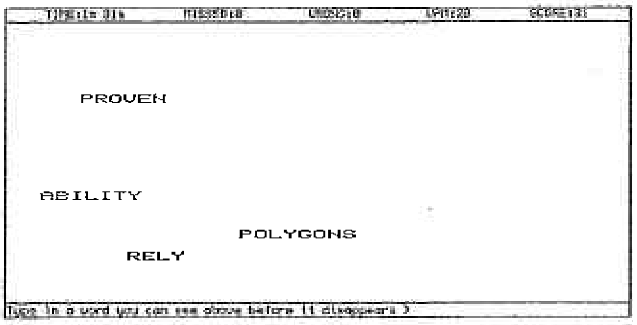
If you'd like a little practice typing and enjoy yourself playing a game at the same time, try out Dilwyn Jones's Wordspot. This free game floats some words or numbers around the screen which you have to recognise and type in as fast as possible. Using the numbers option could be useful for improving your data entry speed using a numeric keypad if your keyboard has one of those.

The better and faster you get, the program increases the difficulty level a bit, in an attempt to help you get even faster. You can set a time limit if you wish and there is a speed control.

The program comes with a plain text word dictionary with about 11,000 words to get you started. It is possible to add more words to the dictionary, and other plain text word lists can be adapted using a small BASIC program supplied.

The program does not require pointer environment, but will run if that is installed.

The program is freeware and can be downloaded from
http://www.dilwyn.me.uk/games/index.html



Type in a word you can see above before it disappears)

WORDBOX

It is with regret that this CD announced in the last issue has had to be temporarily withdrawn, as certain copyright issues came to light with some of the content of the Encyclopaedia section. Sorting this out is taking far more time than it should and I have decided to withdraw the CD until I can be certain of clearing this up. In the meantime, I continue to work on adding more programs and general content, so when it is eventually relaunched in a few months it ought to be even bigger and better.

PCB DESIGN

This highly respected PCB design program has now been further enhanced by its author, Malcolm Lear. The program now stands at version 7.18. The biggest change is the option to import a bitmap, the best way to place a company logo on a printed circuit board. Malcolm says this is an option normally only found on more expensive PCB designing software.

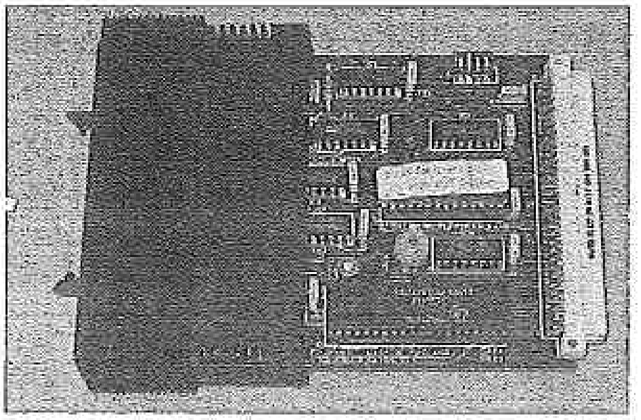
The latest version can be downloaded free from
http://www.dilwyn.me.uk/graphics/index.html

SINCLAIR/MP DISK INTERFACE MANUAL

If anyone has ever bought a second user QL interface only to find it came without a manual, it can be a frustrating experience trying to get it working when you know nothing about it. As the Sinclair/Micro Peripherals interface differs greatly from most QL disk interfaces - it doesn't use the device name FLP and has a set of DIP switch settings - even a "generic" disk interface manual might not help much. So, with grateful thanks to Rich Mellor for scanning it and sending me a copy, I have now placed a PDF of this disk interface manual onto my website so that anyone buying one of them without a manual can get the disk system up and running more easily.

What I haven't got at the time of writing is the utilities disk which came with the disk interface. If anyone still has one, I'd be very grateful to receive a copy I can archive it for anyone obtaining one of these interfaces with the floppy disk missing.

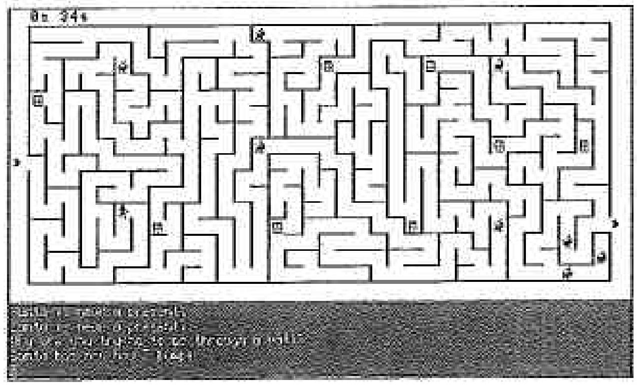
The replacement manual is available to download from
http://www.dilwyn.me.uk/docs/manuals/index.html



SANTA'S MAZE

Just before Christmas I released a QL game with a Christmas storyline. To some extent I now regret it, because I wasted a lot of time over Christ-

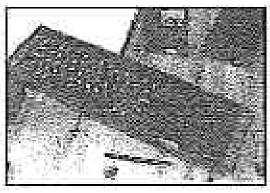
mas (and since then too) playing this game! It's fun to play all the year round, despite being Christmas themed.
Santa has a problem. Quite a severe one. But we won't get into too much detail about that. Instead, the theme of the game is that Christmas is under threat! Santa is beside himself with worry because a mischievous person has sought to sabotage Christmas for little children (and grown up ones!) by letting the reindeer loose in a horrible maze which has only one clear way though, and to top it all, the same person has scattered the children's presents throughout the maze! Now you can see why Santa was worried, how can he face the children without the traditional presents and reindeer?
The prime suspect of course was the QL Today Editor. But he has an alibi - he was too busy writing about Quanta at the time. The other prime suspect was the Quanta Editor, but according to his local police he was out of the country at the time (something to do with making the QL Today Editor nearly choke on his muesli...don't ask!).
So suspicion falls on YOU, dear reader. You can remove all suspicion from yourself by helping guide Santa through the maze to round up the reindeer and pick up all the gifts scattered throughout the maze, and help Santa to escape from the exit of the maze once he has managed to collect everything. Your reward is to rescue Christmas for all the children.
The game is freeware and can be downloaded from:
http://www.dilwyn.me.uk/games/index.html



INFORM AND INFOCOM ADVENTURE PACKAGES FOR QL
I was recently told that these adventure game packages available from my website did not work properly - the executable programs could not be used because of file headers going missing or being damaged.
The Inform package by Luke Roberts lets you create Infocom-style text adventure games for a QL. The Infocom Routines package lets you play and even cheat with Infocom games, up to version 5. There's also three disks of free games to use with the adventure games interpreter.
I've now repaired the files in the zips on the games page on my website. They can be downloaded from
http://www.dilwyn.me.uk/games/index.html

QALENDAR
That most desirable of QL accessories, the QaLendar for 2011 is now available to download from:
http://www.dilwyn.me.uk/gen/calendar/ calendar.html
The calendar is available as either a Word .doc file for Office 2007/10 (if you wish to edit it) or in a more compact PDF format.

March 2011



Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12

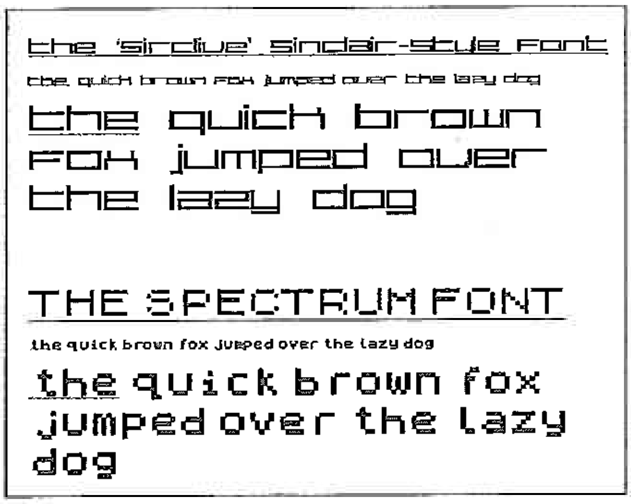
LINUX QL ON A STICK REQUEST
I have been asked about the possibilities of a QL On A Stick version for Linux, or even a version which could be used on both Linux and Windows. The current version of QL On A Stick is for Windows only.
Unfortunately, I have no knowledge whatsoever of Linux and wouldn't know where to start.
I presume that what was asked for would involve the CD or pen drive having:
(1) QLay for Linux
(2) uQLx
(3) QPC2 demo version with WINE
(4) Any other emulators or utilities considered appropriate
Would anyone be willing and able to help us with this? I'll gladly send a CD copy of the Windows version to anyone willing to help.

QLAY2 AND QL2K EMULATORS
In case it helps anyone with using the QLay2 and QL2K emulators, I've posted a couple of reviews of the emulators as PDF files on my website's documents section.
http://www.dilwyn.me.uk/docs/qlay/index.html

The QLay 2 article in particular gives details of how to use the tools programs for transferring files to and from floppy disk, as QLay2 and QL2K have no FLP device.
The articles were originally published in QL Today back in 2007/8.

SINCLAIR FONT

Thanks to Tim Swenson, I have managed to obtain a True Type font with characters which look like the font Sinclair used for their company logo. This might prove useful to anyone recreating historical documents and so on. The font is called 'SirClive' and is described as 'A tribute to the man who REALLY started home computing'.
TrueType, Type 1 and QL Proforma (e.g. for Line Design) versions are available. There is also a similar font, called SFSquareHead which is similar to, but not quite the same as, the 'SirClive' font. There is also a Spectrum-style font on the same page. If anyone knows of a TrueType font which looks like the QL fonts I would love to be able to add such a font to this collection.
The font is available to download free from the Fonts page on my website
http://www.dilwyn.me.uk/fonts/index.html


the 'sirclive' sinclair-style font
the quick brown fox jumped over the lazy dog
the quick brown fox jumped over the lazy dog

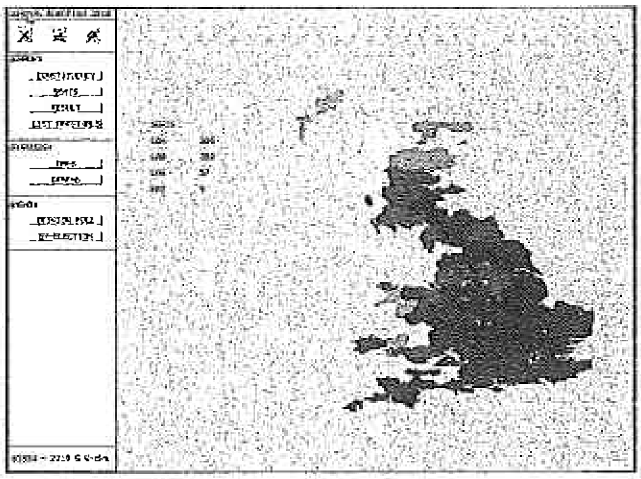
THE SPECTRUM FONT
the quick brown fox Jumped over the lazy dog
the quick brown fox jumped over the lazy dog

George Gwilt Website Change

George Gwilt has announced that his programs can now be downloaded from:
http://gwiltprogs.info/
The previous address run by ukonline is obsolete.
He has also updated two programs:
"A recent update of GDLIB which is a library of routines for Assembly programs. The update consists of an improvement in the routine allowing a non PE program to be moved by pressing F9 and following the instructions."
"A new version of GWASS allows the input of octal numbers by using the prefix AT (@).
Thus 46 can now be entered as one of: 46 $2E ·56 %101110"

UK General Election 2010

Somewhat belatedly Just Words! has released the 2010 version of its General Election analysis program. The interactive map had to be completely rewritten and this was done using new plotting techniques to give the most accurate and detailed map yet.
GENERAL ELECTION 2010 gives the results for the main parties in England, Wales and Scotland. There are numerous analytical possibilities and extensive graphics including interactive political maps. It comes in two versions, non pointer QL colour and pointer GD2 colour. The latter requires a minimum screen resolution of 800 x 600 pixels.
The 2005 version of the program has been added to the archive of every UK General Election since 1983.
The program can be downloaded from the Freeware Downloads page of the Just Words! website:
http://members.multimania.co.uk/geoffwicks/justwords.htm



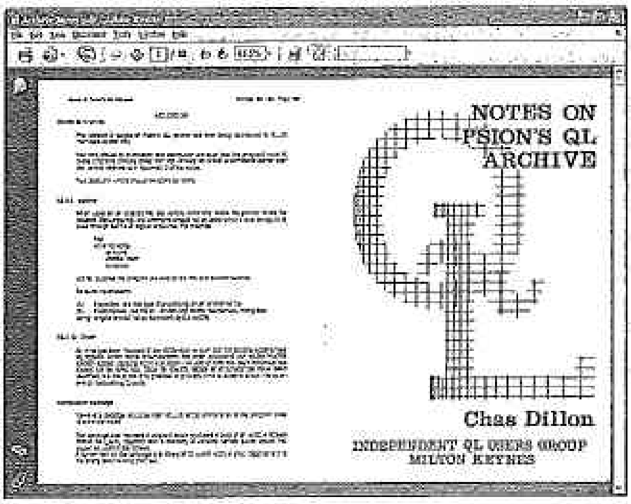
TRON Light Cycles

Urs König writes:
"For the first time since the QL went into oblivion (must have been 1986 or so) the industry had mercy with the QL community and the guys in charge let it happen. Walt Disney Productions decided to allow a QL game release with their recent blockbuster motion picture TRON LEGACY 3D, a sequel of their 1982 cult film TRON. The QL game is called TRON Light Cycles. There's a video with a download link on this QL game."
http://www.youtube.com/watch?v=l_Q0sTLhqKw

QUANTA News

John Gilpin has managed to obtain and scan a very early paper publication, originally published by Quanta. This is Chas Dillon's "Notes On Psion QL Archive". We are grateful to former Quanta

Software Controller John Gregory for lending us his copy to scan so that we may preserve this important document.
It is a 52 page document with extensive notes on using and programming the Archive database. It includes example listings and despite the age of the document, it will quite probably prove very useful to members who would like to learn to use Archive.
We intend to make this available once more for our members through the Quanta Software Library. Quanta members can request copies of Library packages be sent to them by email (reasonable quantities only, otherwise the Library CD is available to those wishing to obtain larger numbers of Library disks) while we await the completion of the Members Section of the Quanta website. The Librarian's email address is available inside the front cover of Quanta magazine.


NOTES ON PSION'S QL ARCHIVE

Chas Dillon

INDEPENDENT QL USERS GROUP
MILTON KEYNES

London Subgroup

Malcolm Cadman writes:
For 2011 we have agreed to move to 8 Meetings a Year. As follows:
1 - February
2 - March
3 - April
4 - May
5 - June
Break - July and August - Summer
6 - September
7 - October
8 - November
Break - December 2011 and January 2012 - Winter
The London Subgroup meetings are held between 2.00 pm and 5.00 pm on the second Sunday of the month in the basement of the Borough Welsh Congregational Chapel, 90 Southwark Bridge Road, London SE 1.

WEB NEWS

ERGON

Davide Santachiara has given a reminder about a change to the Ergon website:
"Since geocities was closed all the Ergon stuff is now held on
www.sinclairql.it
Specifically the utility web page containing Masterbasic is:
http://www.sinclairql.it/qlutilities.htm"

DUNBAR WIKI

Norman Dunbar reports that he has had to change write access to his Wiki because of problems through hacking. He writes:
"Just a quick note to explain a change to the qdosmsq.dunbar-it.co.uk "Qdos Internals" Wiki.
It was hacked by some numpty who registered and created a couple of pages of spam advertising for "free stuff". I have removed the pages and deleted the user in question (he/she/it wasn't one of us!) and removed the spam pages.
I've now had to make a few changes to the Wiki so that when new users register they only have read access. Anyone wishing to make changes will need to request write access until further notice.
You do not need to register to read the contents of the Wiki, only to update.
Existing users who have had write access until now, STILL have write access. It's only new registrations that are affected.
Sorry about this.
I get an email automatically when a new registration takes place, if I recognise the user's name or email as a trusted person from this list, I'll happily give write access without needing to be prompted."

CLARIFICATION

Some misunderstandings have occurred from the section of the editorial in the last issue on the problems we sometimes have with copy. These were intended as general comments and not addressed to any individual contributor.
It is perfectly acceptable to send your copy as a Microsoft Word document with illustrations embedded in the text to indicate the layout and placing of these. We ask you to also send them as separate files to avoid any loss of reproduction quality.

News

25 Years - Part 6

by Tony Tebby

Gazing into the future

The past 25 years have brought a more or less continuous decline in software performance (efficiency, reliability, predictability) coupled with ballooning software sizes and development costs. What does the future have in store?

The view from 1984

Perspectives on fifth generation computing[23], by Brian R Gaines, is a retrospective view of the major developments in computing. It was published the year that the QL was launched. The paper includes Withington's 1974 futurology, slightly modified to fit the circumstances.

Withington's analysis describes the first three generations of computers and predicts the fourth and fifth.

1. 1953–1958 Gee whiz – computers can do anything, technological curiosities: thermionic valves and mercury delay lines.
2. 1958–1966 Paper pushers, business machines: transistors and magnetic core memory.
3. 1966–1974 Communicators: LSI and teletypes, operating systems and communications protocols.
4. 1974–1982 "Information custodians": central databases and satellite timesharing systems
5. 1982–xxxx "Action aids": distributed intelligent systems

References to the fourth generation had appeared from the mid 1960s onwards: Withington's bold stoke was to predict that it was about to appear in 1974, the same year as he published. It did not.

The fourth generation did appear later, in 1990, in the form of Plan 9 (see Vol.14, I.4, Page 25) and, possibly, other systems, but, by then, it had already been overtaken by the un-planned, un-generational, PC.

Seven years after Withington's article, in 1981, with the fourth generation still nowhere in sight, a Japanese group announced their intention of skipping the disappearing fourth generation and going straight for Withington's fifth generation: this was the Fifth Generation Project which shook up the computing world.

Gaines' bold stroke in 1984, was to announce that the fifth generation was already in place, which it was not, except as vague, impractical concepts in research laboratories isolated from reality. His vision of the future was more of the same.

Gaines' concepts are firmly anchored in a 1960s view of computing architectures: time-sharing clusters or clouds, usually with some form of centralised administration, serving terminals. In his 1984 paper, very little weight is given to personal computers and workstations and none at all to embedded systems both of which were emergent. His "distributed intelligent systems" did not consider the possibility of distributing the intelligence to independent units that might have only limited communications, as has been the dominant form of computing since the 1980s, but clung to the 1960s idea of interdependent units working closely together.

Time sharing sharing systems serving terminals had some advantages over totally independent systems. Within the "original" computing environment, university campuses, any student or any member of staff could access "their" data by logging in to any terminal – they did not all need their own workstation. The terminals had negligible intelligence so, quite naturally they had to share the central processing power as well. Since the early 1980s, however, the main requirement for sharing IT resources has become sharing information, not processing power.

Is applying the campus time-sharing architecture to all computing systems just another case of "generalising from one example" (see X Window System, Vol.14, I.3, Page 36) where an architecture designed for one application (university campuses) was applied to other application regardless of the individual requirements?

23 http://pages.cpsc.ucalgary.ca/~gaines/reports/MFIT/OSIT84/OSIT84.pdf

Not at all. By 1984 it should have been obvious that, even for a campus environment, having terminals time-sharing a cluster or cloud of processing units was not a sensible approach where the distributed processing power in the terminals would far outweighed the processing power of the central cluster. The original campus time-sharing architecture was obsolescent even in its original environment.

In 1984, the new requirement was independent processing and, to a lesser extent, the central management of shared and private data, but the entire edifice of the 1960s theories was based on the now irrelevant concept of "competing processes" on shared computers. From 1984, large system architectures should have moved towards independent processing, on independent machines, accessing shared data. Furthermore, the new requirement for shared data was for "transactional" (asynchronous) access to long term data storage, giving rise to a totally different set of problems to those addressed by the seriously flawed 1960s "synchronisation" theories.

The critical difference between the new approach required and the old approach is that the 1960s theories were algorithm based and conflated data with processing. This conflation reached its apogee in object oriented programming where data and processing of that data are made indivisible. For an independent processing / shared data approach, the data, and the management of that data, must be rigorously separated from the processing of that data. This already tends to happen by default, as it is often the only practical approach. It should be done by design.

The view from Berkeley (California) 2008

Parallel Computing: A View From Berkeley[24] sets out the Berkeley view of the future of mainstream computing. The title is slightly ambiguous. Does the title imply that the paper is only concerned with the future of parallel computing or does it imply that parallel computing is the future?

In a world where the arrival of the massively parallel fifth generation of computing had been announced 24 years before but was still not even vaguely on the horizon, this paper takes the position that the only way forward is parallel processing.

The importance of this paper is not in the content, but the in extent to which, like Withington's and Gaines' papers, it has been adopted, quoted and recycled within the computer science community.

The paper sets out its stall in two ways, one approach considers "conventional wisdoms" and how these need to change and the other considers the state of the art and the future of computer hardware.

Berkeley's conventional wisdoms

The paper sets out a number of "old conventional wisdoms" on system design with corresponding "new conventional wisdoms". These terms are very strange: for "old conventional wisdoms" a better term would be "parody of reality" and the "new conventional wisdoms" would be better described as "old dogma".

The first few "conventional wisdoms" merely repeat the 1960s dogma that processors cannot be improved significantly. Then there is a group of "conventional wisdoms" (CWs) that represent a total denial of reality.

9. Old CW: Don't bother parallelizing your application, as you can wait a little while for a faster sequential computer
9. New CW: A faster sequential computer won't arrive for a long time
11. Old CW: Increasing clock frequency is the primary method for improving processor performance
11. New CW: Increasing parallelism is the primary method for improving processor performance
12. Old CW: Less than linear scaling for a multiprocessing application is a failure
12. New CW: Any speedup via parallelism is a success

These are surreal.

24 http://www.cs.nyu.edu/srg/talks/BerkeleyView.pdf

The old Berkeley conventional wisdom, for the past 40 years, was 'don't bother about making your application efficient, you can always parallel the processors for more speed', whereas, in reality, rapidly increasing processor speed has been the practical means of compensating for rapidly declining software performance. In these conventional wisdoms, they are claiming that parallel processing, the fundamental basis of the fourth generation which was due to arrive about 1974, is THE NEW IDEA for 2008.

Berkeley's start of the art

It would be too easy to dismiss the View From Berkeley as merely an attempt to put a 21st century gloss on a 1960s tract in favour of parallel processing, but there is more to it than that.

The paper starts off with a graph showing that, up to 2002, processor ***benchmark*** performance doubled every 18 months but that the rate is now much lower. This corresponds roughly to the introduction of multi core processors, first for RISC processors, then for CISC processors. This shows that the generalisation of these multi core processors did not maintain the upward trend in performance.

This is followed by a glimpse of reality. "Parallel processing is nothing new. In the past, research in parallel processing was driven by new breakthroughs which opened design space, with uniprocessors always prevailing in the end." This seems to accept that forty years of systems development concentrated almost entirely on symmetric multiprocessing had totally failed to displace single processor systems as the dominant form of computing.

So Berkeley's state of the art in 2008 was that processor performance was running into a brick wall and parallel processing had hit a brick wall years before.

Berkeley's hardware future

The whole of Berkeley's hardware future is based on their 10th "conventional wisdom". Their conclusion from this is that the future is in massively multi core processors with "thousands of cores on chip".

> 10. Old CW: Uniprocessor performance doubles every 18 months
>
> 10. New CW: Power Wall + Memory Wall + ILP Wall = Brick Wall. Uniprocessor performance now only doubles roughly every five years

Once again, the "Old Conventional Wisdom" is not – it is just an observation that was true for strictly defined and not very useful conditions between about 1984 and 2002. The "New Conventional Wisdom" is just an idiocy plus a deliberately misleading statement.

It is true that the graph of processor benchmark performance shows that the performance used to double every 18 months but now the improvement rate is much slower. The misleading statement, however, dissimulates two important factors. The first is that the gap between real performance and benchmark performance has widened over this period and the second is that the tail end of the graph represents the introduction of multi core processors, not the stalling of single processor performance.

The idiocy shows how even the most stupid propositions gain an air of veracity if they are repeated often enough. The well known, frequently cited, and totally wrong "Power Wall + Memory Wall + ILP Wall = Brick Wall" is raised as an argument in favour of multi core processors.

The "Power Wall" is a ridiculous simplification of the problems of reducing the cycle time of the processor core. Power is a factor, but in the 1970s, IBM was already delivering water-cooled processors, so this is not a barrier, it is just a one of a whole series of physical limitations on the processor cycle time. These limitations have, however, not stopped processor core execution speeds increasing far faster than overall processor execution speeds as processors have become more and more limited by the "Memory Wall".

The "ILP Wall" (instruction level parallelism) simply does not exist. The gains that can be obtained using ILP are extremely limited but these gains are proportional to the processor clock speed. If a given ILP strategy can double the effective speed of a 1 GHz core to 2 GHz (two instructions per cycle), then the same ILP strategy will double the effective speed of a 2 GHz core to 4 GHz. At the processor core level, ILP is purely scalable but not very useful: at best ILP is just a quick patch to get around the problems of primitive instruction sets with excess serialisation. The reason that ILP does not yield its theoretical scalability in real processors is that an increase in instruction execution speed does not translate directly into an increase in overall processor speed because, since the early 1980s, processors have been running into the 'Memory Wall'.

The 'Memory Wall' is a real problem. Elsewhere in the paper the author states "The gap between memory access speed and processor speed increases by 50% per year". Over this period, main memory access times have increased by about a factor of 5 and memory bandwidth has increased even more with wider buses and larger burst accesses, but this is still much less than the 10,000 times increase in processor core speeds.

The most important thing about the 'Memory Wall' is that it imposes the same ultimate performance limit ***regardless of the number or speed of the processor cores.*** It does not add to the core performance limits, as would be suggested by the ridiculous 'Brick Wall' formula, it overrides them.

The 'Memory Wall' is the multi core killer.

The paper has a big title 'Multicore Not the Solution' that acknowledges the failure of multi core technology (which had ran into the memory wall even before it even became widely available) but proposes adopting it in an extreme form: '1000s of cores per chip', totally ignoring the memory access problems. It is difficult to see any logic in this.

Berkeley's software future

This paper does mark two distinct 'advances' on the 1960s approach to parallel processing.

By the end of the 1960s, the computer science establishment believed that it had the whole solution to parallel processing problems, largely based on the twin concepts of synchronisation and symmetric multiprocessing. Later, Berkeley was one of the prime movers in putting synchronisation back into Unix for symmetric multiprocessing. This paper is, maybe, a mea culpa, as it advocates finding alternatives to synchronisation.

> Locking (synchronisation) is notoriously difficult to program for many reasons (deadlock, noncomposability). Locking is also wasteful in that it busy-waits or uses interrupts/context switches.

So, in plainspeak, synchronisation increases software development cost, while making systems less reliable and less efficient. It has just taken Berkeley 40 years to realise this fundamental truth.

> Switching from sequential programming to moderately parallel programming brings with it great difficulty without much improvement in power-performance. Thus this paper's position is that we desperately need a new solution for parallel hardware and software.

Finally accepting, after 40 years of development of the 1960s multiprocessing theories for parallel execution, that a new solution is desperately needed might be considered some form of advance, even if the proposal is to wind the clock back to 1962 and start again in the same direction. Since it would appear that there is no intention of challenging the false premisses on which the whole multiprocessing edifice was built, it would seem likely that the same mistakes will be repeated all over again.

The view from cloud-cuckoo land

Since the early 1960s one of the unshakeable dogmas underlying systems architecture development has been that single processor systems were doomed as they could never meet increasing performance expectations and the only way forward was to distribute the processing across thousands of processors using a well organised symmetric multiprocessing model. In practice, processing has been been distributed but, instead of all processing being carried out on organised systems with thousands of processors working in parallel, as envisaged in the 1960s, the bulk of processing is now carried out by a totally anarchic mass of independent PCs, embedded computers and hand-helds.

From a computer science point of view, this is wrong. From a user perspective, the current anarchy is certainly better than the fourth generation / fifth generation horror that it has successfully kept at bay. The challenge is not to bring the anarchy under control, it is to make it work better by working with it rather than against it.

Improving software quality (efficiency, predictability, reliability) on the individual computers would be a good start.

Wirth's law on bloatware is not theoretically justified, it is merely an observation which seems to be as true today as when it was formulated. There is no justification at all for the ballooning costs, size, and inefficiency that has thrown away the performance that has been gained by the extraordinary development of computer hardware over the past 25 years.

The computer science world has taken perverse pride in promoting software inefficiency as an implicit aim: anyone who challenges this is treated with utter disdain. For 25 years "you can always get a more powerful computer" has been the "conventional wisdom": efficiency is just a dirty word used by grubby little hackers.

Recovering the processing power gains that have been achieved since 1984 is, however, not necessarily the most serious long term problem to be addressed. On the one hand, workstation and server hardware architectures have been compromised by trying to fit in with 1960s symmetric multiprocessing, on the other hand, system architectures have been compromised by a mindset on realising the fourth generation or even the fifth generation.

In 25 years, this 1960s mindset appears to have become only more rigid so that, to achieve any significant deployment in the short term, any new development will have to be within the existing archaic hardware and infrastructure architectures.

The cheapest route to long term gains in system performance is not increasing clock speeds or paralleling processors, it is improving the software quality. This cannot be done retrospectively: the quality must be built in by designing for reliability, predictability and efficiency. Improved software quality brings with it reduced hardware demands, lower power consumption, reduced size, longer battery life as well as more predictable performance: are these really as unimportant as conventional computer science would have us believe?

The argument against a revolution in systems software is that compatibility must be maintained at all cost. This is nonsense, although there have been outstanding failures to revolutionise a market (the Apple Mac into the PC market in 1984 and Linux in the PC market ten years later), the problem was that the "new" systems were promoted as being different (which they were, superficially) rather than being significantly better for the target applications (which they were not). On the other hand, the PCs of 25 years ago were completely incompatible with mainframe computers, but they wiped them out. Palm Pilots were incompatible with everything else but they were a great success. The pocket computers that wiped out the Palm Pilot were incompatible with anything that went before.

It is possible to make systems that are significantly better, but this cannot be done without making them significantly different, built in a significantly different way. We know it can be done. We can be sure that most of those setting out to do it will just recycle all the old junk theories and end up with just another Unix. Who will be bold enough to break the cycle?

In the 1970s and early 1980s, the development time for an operating system used to be about 1 year. The development of an application program (a word processor, for example) used to be about 1 year. The development of a programming method used to be about 1 year.

It is true that we ask more of applications now than 25 years ago. But it is also true that we should be starting from a more advanced position with more advanced tools so that the core development should be much faster. The additional functions that we have come to expect do not have much effect on the core structure, so they can be developed in parallel. It should, therefore, be possible with a modest outlay (about 0.01% of the estimated true development cost of Linux) to develop, within one year, a complete system with all the applications required of a data server, workstation or personal computer and with a responsiveness and reliability that does not make you wish for your 1980s system.

There are three problems with this scenario.

The first is that the development would not be starting from a significantly more advanced position than 25 years ago because there has been almost no significant advance – just lots of hot air and recycling. It would be possible to leverage some peripheral technologies such as 16 bit (or more) character sets, glyph definitions and image and video file formats. However, unlike 25 years ago, the system would be constrained by not be able to set its own standards but having to cope with "industry standards" that have gone completely off the rails, making development significantly more difficult.

The second is that there do not seem to be any tools that are significantly more advanced than those available 25 years ago: if they do exist, it would probably take longer to track them down and evaluate them than to create suitable tools from scratch.

The next problem is building a market outside the established workstation market. There is no point in attacking established systems head-on in their own market, so systems must be targeted at new applications and markets that will be made possible by a "quantum leap" in price/performance, something that has been done many times over the past 25 years in many domains, and can be done again, even in the computer world. Already, there is visible discontent with the feature overloading that even cell phones now suffer from and there is a growing demand for a more lightweight approach ("less is more") in the developed world. More radically, the dramatic reduction in hardware costs and power consumption and dramatic improvement in systems accessibility that would come from adopting a more rational software technology opens up vast possibilities on the other side of the digital divide.

Finally there is the problem of finding the funding, not so much for the development, but for the marketing. There, the problem is that there are very few people of the stature of Sir Clive Sinclair. He looked at Unix and saw no justification for its appalling performance and fragility and he could not see why an equivalent system could not be implemented on a hobby computer. He bet his own money and was proved to be right. In the computer science world, challenging orthodoxy and being proved wrong is a forgiveable aberration, challenging orthodoxy and being proved right is totally unforgivable.

A Triangular Exercise

by George Gwilt

I recently wrote a PE program allowing a user to pick three points, A, B and C which were displayed on the screen with lines joining them. Then the angle ABC was calculated and shown. This seems simple enough. This is what I did.

Finding the Points

The first step was to find the positions of A, B and C. This was done by allowing the user to move the pointer to the position he wanted to choose and then clicking the mouse button. This was achieved by reading the pointer with the function `BRPTR`

The actual instruction was:

```
    ptr = BRPTR(10,oldk,wwd)
```

The number 10 causes the pointer's position to be returned either if the pointer has moved from the original position, oldk, or if the mouse key was pressed. The item "wwd" is the address of the window working definition.

The function `BRPTR` is part of TurboPTR but you can use the trap #3 routine `IOP.RPTR` to do the same thing in assembler if you want to do it that way.

Printing the Letters

When the user has chosen a position, the appropriate letter is printed at that position. Since the position of the pointer is given in absolute co-ordinates these have to be corrected to the relevant co-ordinates before printing can occur. The adjustment is made by subtracting the absolute position of the top left of the main window from the absolute pointer position and then, further, subtracting the origin of the sub-window in which the points appear. The information is found easily enough from the window working definition.

Once the position of the point is found, in pixel co-ordinates, the appropriate letter, A, B or C can be printed there. There is one problem here though. If the point is chosen too near the edge of the sub-window an out of range error will occur when I try and position the cursor. Thus, since I decided to print the letters so that their midpoint coincided with the point in question, I restricted the search for points to an area smaller than the sub-window in which the search is taking place by a margin of three pixels horizontally and five vertically at each edge.

I did this by means of the procedure `SET_PTR` which sets the pointer to a particular position. This procedure is based on `IOP.SPTR`

Drawing the Lines

An interesting problem arose when I tried to print lines from A to B and from B to C. I assume it is obvious that these should be drawn by using the graphics procedure `LINE`. However, although the keyword `CURSOR` and the underlying `SD_GCUR` allow the pixel positioning of the cursor relative to a graphics point, there seems no easy means of finding the graphics coordinate of a given pixel position.

The easiest approach seemed to me to set `SCALE h,0,0` where h is the height of the window in pixels. This sets the graphics scale so that the height of the window is the same in pixels or graphics co-ordinates. The pixel y co-ordinate is now just the height less the graphics y co-ordinate

However, the corresponding pixel x co-ordinate is not so easily determined. This stems from the fact that in an original QL the screen pixels are oblong, not square. For example, in discussing graphics, Jan Jones in her book on SuperBASIC suggests that

```
    OPEN #3,SCR_137x100
```

will produce a square window.

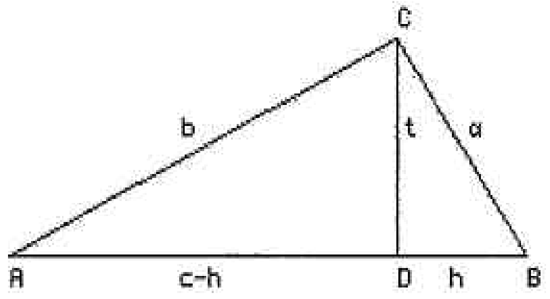
This implies that 137 horizontal pixels give the same distance on the screen as 100 vertical ones. So, how do we tell how many horizontal pixel units should we allow for a given number of horizontal graphics units?

A recent e-mail to the QL group from Marcel Kilgus solved the problem, at any rate for SMSQ/E users. The decoded form of the e-mail is this. One of the entries in system variables is an address at `SYS_VARS + $C4`. This is called `sys_clnk` which is a pointer to the console linkage block. You can see definitions of the contents of this large block in the source code of SMSQ/E. The file to look at is called keys_con. Perhaps the information appears elsewhere but either I have missed it or it doesn't. Anyway, the item at `sys_clnk+$14A` is called `pt_asprt` and is the floating point value of what is termed the pixel aspect ratio. This value, let's call it par, can be used to find the pixel value of the x graphic co-ordinate, let's call this gx.

The pixel x co-ordinate then is gx/par.

Determining the Angle

The last step in my program was to find the angle ABC. The following diagram shows the



points A,B and C in a triangle.

The line CD is the perpendicular from C to the line AB. The lower case letters are lengths. Thus a, b and c are the lengths of the lines of the triangle ABC opposite the points A, B and C. The height of the perpendicular CD is t.

By Pythagoras we have:

```
1. t^2 = a^2 - h^2
2. t^2 = b^2 - (c - h)^2
```

From these by eliminating t we derive:

```
3. a^2 = b^2 - c^2 + 2*c*h
```

Thus:

```
4. h = (a^2 + c^2 - b^2)/(2*c)
```

The angle ABC in degrees is then:

```
 DEG(ACOS(h/a)
```

Letter-Box

Dilwyn Jones writes: "Futura"

Thank you very much for the articles about QL history from Tony Tebby and Urs König. Reading them has been a pure joy and brought back many pleasant memories of using my QL back in the 1980s. If I'd known then that the QL was still going to be around, that I'd still have a working QL and that my main 'QL' would have been an emulator running on a PC (which runs even faster than the original QL) in 2011 I think I'd have written down a lot more for posterity back then.

I was particularly fascinated by Urs König's tale of acquiring the Sandy Futura motherboards. I remember seeing the adverts for this computer back in the mid 1980s and was very sad that it didn't really make it to the market, as it would have been, as Urs put it, a bit of a dream machine back then.

Following the article, I was lucky enough to get to talk to an ex-Sandy employee about the Futura. He said that despite the impressive adverts for the machine it was never really a high priority at Sandy, because they had more profitable sales from the QL interfaces they sold, and the sale of a fairly large number of QLs and QL boards they acquired cheaply at that time - I wonder if readers remember the QXT-640 re-cased QL system they produced for a while, for example?

Apparently the name Futura had been chosen by magazine readers in a competition.
There were apparently some issues with the video - whether that explains what Urs mentioned about the graphics card, I don't know.
In view of this, from what I was told, Tony and Arnie at Sandy (those in charge of the project) decided the machine would not be successful because it was too expensive to produce, and the margin on it would be less than selling QLs and their accessories. Later, they repurposed the Futura design as something a bit different and of course eventually Sandy became Power Computing and concentrated on other systems in due course. I was also told that at one point Sandy had intended to help Alan Miles and Bruce Gordon of MGT (Miles Gordon Technology) to produce the SAM Coupe computer, as they felt that had the potential for a better market. That took too long and Tony and Arnie backed away and left that to MGT, feeling perhaps that the time had passed for a 'super Spectrum' back then.
If any complete Futuras were ever sold, it seems that the total number in the hands of users was probably no more than ten.
I hope Urs does get his 'dream machine' up and running in time and if does manage to do so, that he writes up his experiences with it for us to see what we missed out on in those early days!

George Gwilt writes: "Letter to QL Today re Norman Dunbar's Article Easy PEasy Part 2"
I am most grateful to Norman in his article Easy PEasy Part 2 for having corrected and improved the code and comments which appear in my example EX0_ASM. However, there are two comments I would make.
The first concerns the statement towards the middle of page 34. 'This code will not return unless an action routine sets D0 with an error code or sets D4 with an action number.' It is true that the code will return if at least one of D0 and D4 is non-zero. However, the code might return as a result of an event key being pressed and not as a result of an action routine. If an event key is not set to select a loose item then pressing it will cause an event and the code reading the pointer will return with D4 non-zero. If that key does select a loose item then whether or not a return occurs depends on the action routine for that loose item.
As it happens, in EX0 the event keys for both move and resize select their respective loose items. Furthermore neither of the action routines causes an exit from the reading loop. The moving and resizing take place entirely within the action routines. As a result of that in EX0 the code checking for a move or size event is superfluous because you could never arrive there. However, that code would have been needed if it had been deemed necessary for the event to be activated by an event key which did not select the appropriate loose item.
The key F1 causes a help event so pressing it will cause an exit from the reading loop. However, since that event is not checked in the code at no_err, the program simply branches back to the reading loop. You can perhaps detect this happening by placing the pointer on a loose item then pressing and holding down F1. You should then see the pointer disappear.
The second comment refers to Norman's implied suggestion on page 36 that there might sometime be a supplied routine to perform a resize as there is for move and sleep. There are several reasons why it is not practical to produce a single routine for resizing.
The first reason concerns the position of the move loose item. In EX0 it is at the top right corner of the window. There are three other corners where it might have been placed. In each of these cases there would have to be a difference in the coding just before and just after 'resze' on page 38 if it is the rule that the opposite corner remains stationary. When a resize takes place it is either the right or left vertical edge that moves and it is either the top or bottom horizontal edge that moves. If it is the left vertical edge that moves, then we find the x-value of the new position of the pointer by subtracting the change in size from the original position. Otherwise the original position is unchanged. Similarly if the top edge is moved we subtract the change in size from the original y-value.
I suppose it is conceivable that someone might want to put the resize icon in the middle of the window and leave the midpoint of the window unchanged on resizing. This would entail subtracting half the change in size from the x- and y-values to find the new pointer position.
The second reason for not having a general resize subroutine is that in some programs, as happens for example in QD, you might want a greater step in changes of size than the minimum I have chosen for EX0. Again this means a change of coding.
A third reason is that you may want more significant alterations in the window depending on its size. One example of that is a window allowing scroll bars to move the contents if the window is too small. For both x and y therefore you will either have scroll bars or not depending on the relative sizes of window and contents. That is something I would most certainly not want to put in a general resize routine.

PSION As Never Before - Part 2

by Geoff Wicks

In case you are wondering why you missed part 1, you probably have not. It was published 5 years ago (V10 I5 P42) and I did not expect there to be a part 2. Events just before Christmas changed all that.

The Psion programs have remained much the same since the first days of the QL, but in 2006 there were some major changes to the Xchange versions. Marcel Kilgus updated the suite to increase the working area from 512 x 256 to 512 x 512 as well as making a number of improvements to the display. Working independently of him Roger Godley had been adapting the Xchange programs for high resolution GD2 systems. In 2006 I wrote a review of the programs that were then available, stand alone versions of Quill, Abacus and Archive. Roger was still working on a high resolution version of Easel.

Just before Christmas Roger sent me copies of two new programs SILVIA and SOLANO. SILVIA is a portrait version of Abacus and Archive and SOLANO a landscape version of Abacus, Archive and Easel. To run these programs you need a GD2 system with a minimum resolution of 1024 x 768. Anything less and the programs will not load.

I stress this point because initially I fell foul of it. My monitor, once state of the art, is now over 8 years old and is small by today's standards. Both my PC and QPC default to a resolution of 800 x 600. To run Roger's software I have to follow a strict procedure of first increasing the resolution of the PC and then that of QPC.

When I loaded the programs I found myself fumbling at first, but this was not Roger's fault. I no longer use the Xchange programs and I had to do some relearning. At this stage I found the size of the screen in SILVIA disconcerting mainly because of the distance between the menu items at the top and the input details at the bottom. However once I had mastered the relearning process I quickly adjusted to this and found that existing files loaded easily into both SILVIA and SOLANO.

To give you some idea of the size of the SILVIA and SOLANO screens figure 1 shows the two programs alongside the original Xchange. Roger suggests the programs would be most suitable for people who want more information on the screen and to illustrate this I have a spreadsheet and a database.

The spreadsheet is a cut down version of the one I use to monitor my energy use, and shows my weekly energy meter readings over a two year period. In figure 2 you can see the spreadsheet in SILVIA alongside it in the original Xchange. In the original Xchange it shows the meter readings from 1st January 2009 to 19th March 2009; in Marcel's version of Xchange from 1st January 2009 to 10 September 2009; and in SILVIA from 1st January 2009 to 22nd January 2010.

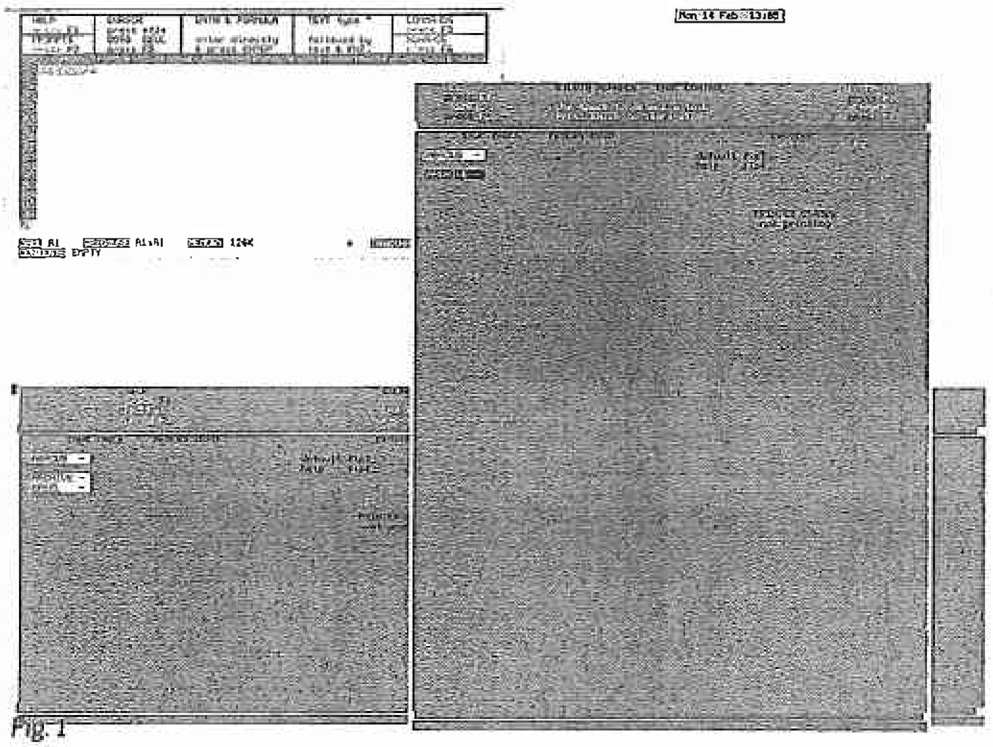
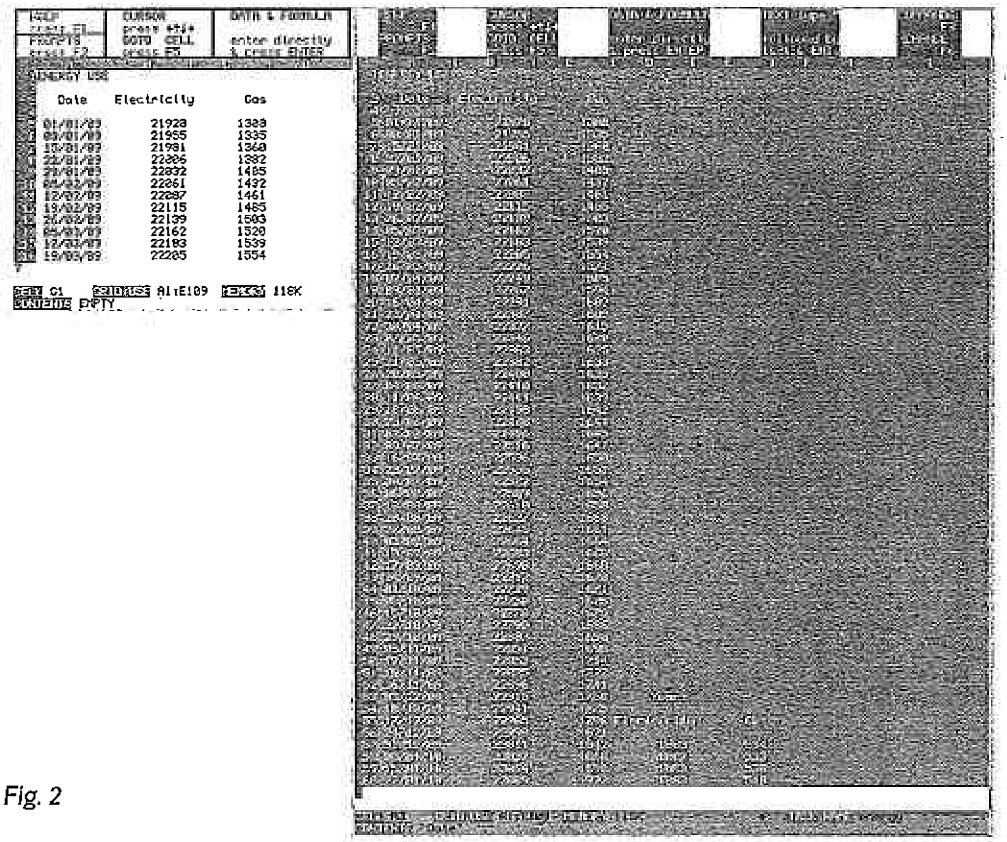


Fig. 1

In SILVIA you can also see more details than just the weekly meter readings. From the 31st December 2009 there is a calculation of the yearly energy use. In other words in SILVIA you can have a yearly view of energy use on one screen without the scrolling you would need to do in the two other ver-



```
HELP          CURSOR         DATA & FORMULA
press F1      press ←↑↓→
PROMPTS       GOTO  CELL     enter directly
press F2      press F5       & press ENTER
ENERGY USE

   Date      Electricity        Gas

 01/01/89        21920          1308
 08/01/89        21955          1335
 15/01/89        21981          1360
 22/01/89        22006          1382
 29/01/89        22032          1405
 05/02/89        22061          1432
 12/02/89        22087          1461
 19/02/89        22115          1485
 26/02/89        22139          1503
 05/03/89        22162          1520
 12/03/89        22183          1539
 19/03/89        22205          1554

CELL C1      GRID USE A1:E109     MEMORY 118K
CONTENTS EMPTY
```

Fig. 2

sions. If I had used the full version of the spreadsheet and not a cut down version you would have seen the energy use each week and every four weeks, and would have been able to make comparisons between the current and the previous year's energy use for the same period without the need for scrolling.

The database, the oldest and largest I have ever written, is an index to some of the photos I have taken during my life. It started life as a Spectrum program before conversion to ASCII text for importing in Archive or a PC database program. Figure 3 shows the partial results for a search of photos taken in the Peak District National Park. Archive finds 141 photos in the database. I have not modified the Archive program for the larger screen of SILVIA, which shows the 17 results that can be displayed in the original Xchange. Had I modified the program for higher resolution screens Marcel's version would have displayed 42 results and SILVIA 61 results. This is typically the sort of database where you want as much information on the screen as possible. In Xchange the information would be displayed over 9 screens; in Marcel's version over 4 screens; and in SILVIA over 3 screens.

Easel is the most difficult of the Psion programs to run in a GD2 environment, which is what you would expect of a graphics program. In the original Xchange data can be entered into Easel and is correctly displayed, but when you try to modify the graphic by, for example, amending text, the screen becomes corrupted. Marcel corrected this in his version of Xchange, but the size of the screen gives some odd looking, ill proportioned graphics. Both versions have difficulty with pie charts.

SOLANO is written for GD2 screens and handles graphics much better than the other two. It does, however, warn that 12 segment pies will not save and suggests you use a different format for saving and then convert back to a pie chart after reloading.

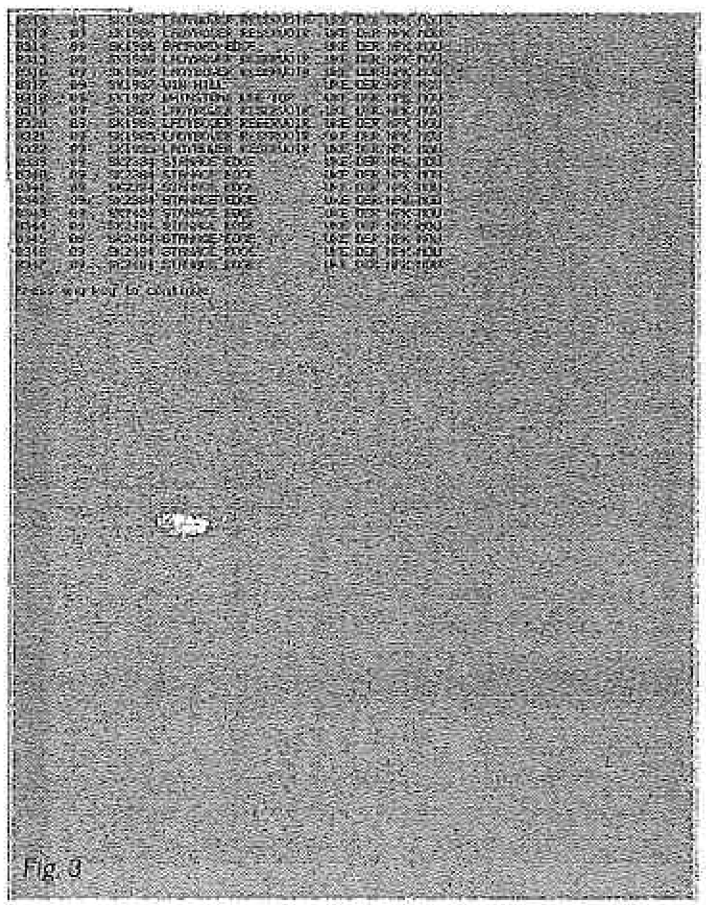
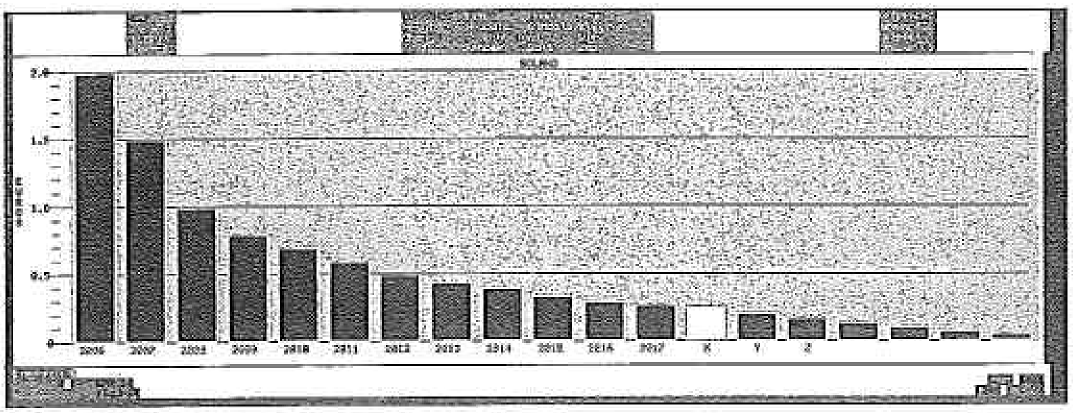
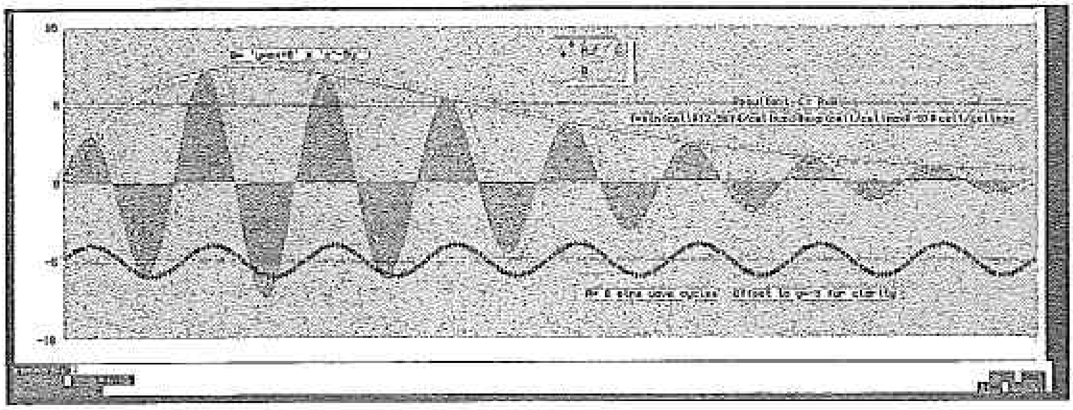


Fig. 3



It would take a long time to do a full review of Easel and I hope someone with an interest in Easel graphics will volunteer to take a deeper look at the program. It would make a good future article for QL Today. To illustrate the capabilities of the program I include two screenshots of Easel graphics that Roger sent me.



The one program that is not included in SILVIA and SOLANO is Quill, although Roger's first modification of the Xchange programs was Quill. In 2006 Marcel told me Quill was the most difficult of the Psion programs to modify. Patching colours was fairly simple, but locating the places to patch them much more complicated especially after modification of the screen size. In practice Roger's version of Quill was more prone to crashing than the original and he no longer wishes it to be available. He also does not wish further circulation of the older versions of Abacus and Archive that I described 5 years ago because he wrote them too long ago to be able to maintain them. SILVIA and SOLANO are the final products of his original work.

Until now Roger's software has only been available from the Quanta Library although it was always his wish that Quanta should also distribute them to non-members. This is something that Quanta has steadfastly refused to do and it has led to some disagreement with Roger. He is also unhappy with the continued inclusion of his older programs in the Quanta Library.

At the moment there are no official channels through which non-members of Quanta can obtain SILVIA and SOLANO, although I am considering putting them on the Just Words! website if I can find the time to rewrite the website and move to another host. It would be the first time that I have opened the website to third party software. As an interim measure I have offered to circulate Roger's programs privately on his behalf. Any reader wishing to have them can contact me via my email address printed on the inside cover of the magazine. I anticipate a download size of just over 300KB. Please include the word "Xchange" in your subject line so that I do not delete your email thinking it to be spam.

A Suprising Thing

by George Gwilt

It all started when a member of a group investigating UQLX exclaimed "LRESPR doesn't work!" I immediately switched to a version of UQLX running on my Apple Mac under an UBUNTU version of LINUX inside VMWare. I was able to report that LRESPR worked for me.

However, later on when the heat of the chase had died down a bit, I looked more closely at the results of using LRESPR. To explain what happened next I must digress slightly.

Slight Digression

One of my programs which I use frequently is **NET_PEEK**. This program analyses the contents of ram, both of the machine on which it is running and on any other machine linked into a network. Analysis includes such things as a list of jobs, a list of channels, the contents of a job's registers and the particulars of any open channel. It also, if running on a 68020+, will disassemble instructions. In short I find it indispensible. Indeed I often have more than one version of **NET_PEEK** running. To load these versions I used to use `EX NET_PEEK` expecting **PROGD$** to be set to the directory containing the program. One day, I was experimenting with C68, for which I had set **PROGD$** to the directory WIN1_C68_. It annoyed me when I typed `EX NET_PEEK` and got the infuriating message 'Not found'. I had to laboriously type `EX WIN1_SYS_NET_PEEK` to get the program to load. I then remembered that another program, much used by me, could be loaded by using **EXEP**. Typing `EXEP QD` always produced another version of QD whatever the setting of **PROGD$**. I determined there and then to make **NET_PEEK** an executable Thing, just like QD.

Two problems had to be solved first though. I wanted to be able to call **NET_PEEK** either as an ordinary executable program by **EX** or as an executable Thing by **EXEP**. To set **NET_PEEK** as a Thing I needed to **LRESPR** it. The first problem was how to program **NET_PEEK** so that it could distinguish between being **LRESPR**d to become a Thing and being loaded by EX. The second problem was how to make **NET_PEEK** into a Thing.

Problem 1 – LRESPR or EX

The first problem is easy to solve. If **NET_PEEK** has been **LRESPR**d the program ID will be that of either master BASIC or that of a daughter BASIC, if SMSQ/E is in operation. If the program ID is zero then **NET_PEEK** has been **LRESPR**d via master BASIC. If the long word just before the position to which A6 points is 'SBAS' then we are in a daughter BASIC. In all other cases it is reasonable to assume that the program has been started by **EX**.
As it happens, I do not allow an **LRESPR** by a daughter BASIC. This is because if the daughter BASIC is removed, then so will the **NET_PEEK Thing**. To give **NET_PEEK** immortality we need master BASIC to do the **LRESPR**ing.

Problem 2 – How to Make NET__PEEK a Thing

The manuals tell us that we need a linkage block defining **NET_PEEK** as a Thing and that we also need a Thing Header.

Linkage Block

The linkage block consists of 42 bytes of information followed by a string containing the name of the Thing. The long word at position $10 is a pointer to the Thing itself. The manual states that the 'linkage block must be in the common heap. .' This block is linked into the Thing list by sms.lthg, the description of which also states that the linkage block 'must be allocated in the common heap . .'

Thing Header

The format of an executable Thing's header is:

Item	Address	Length	Value
THH_FLAG	$00	4 bytes	"THG%"
THH_TYPE	$04	long	1 (for executable code)
THH_HDRS	$08	long	offset to start of header
THH_HDRL	$0C	long	size of header
THH_DATA	$10	long	dataspace
THH_STRT	$14	long	offset to start of code or 0

Notes

1. The offsets are calculated from the start of the Thing Header.
2. If a program does not alter its code it is said to be re-entrant. In this case it is possible to have several instances of the program in force at one time with all of them using the same code. Each of the programs will have its own entry in the Program Table and its own Header followed by the few bytes at the start of the program containing its name. After this comes the dataspace. The alternative is to have mul-

tiple copies of the whole program. The Thing Header supports both these alternatives. For the first, **THH_DATA** contains the length of program to just after its name and **THH_START** points to the single copy of the whole program. In the second case, **THH_DATA** contains the length of the whole program and **THH_START** is set to zero.
Bearing in mind that the Linkage Block had to be in the 'common heap' I decided to put it at the end of my program **NET_PEEK** just after the Thing Header, since presumably the program would be put in the common heap by **LRESPR**.

End of Slight Digression

If you remember, I was just looking into **LRESPR**. Well, I decided to try out **LRESPR** on **NET_PEEK** to make it a Thing. This had worked well on Q40, Q60 and QPC2 to name a few platforms. Without any problems UQLX accepted the **LRESPR** command. However, when I typed `EXEP NET_PEEK` I got an error message. I think it was 'Not found' but I'm not sure. My mind had just then become completely blank. Luckily I was able to load **NET_PEEK** by using **EX**. I used that version to trace through all the Things to make sure that **NET_PEEK** was indeed linked into the Thing list. Since it was in the list I couldn't understand why EXEP did not work. In an attempt to find out I looked more closely at **NET_PEEK**'s linkage block and at its Thing Header. After a bit I noticed that the item **THH_HDRL**, which should have been a relatively small number was incredibly large! I noticed in fact that it was the absolute address of the Linkage Block. What was this doing in the middle of **NET_PEEK**'s Thing Header?
The code at the end of my program was as follows.

```
; This is the NET_PEEK Thing
N_THING   DC.L      "THG%",1
          DC.L      HEADZ-N_THING      ; Pointer to header
          DC.L      PRS-HEADZ          ; Size of header
          DC.L      D_SPACE            ; Amount of dataspace
          DC.L      STARTA-N_THING     ; Pointer to start of code

; This is the linkage block
TLINK     DCB.W     19,0
          DC.L      "1.00"
          HED1      <"NET_PEEK">,TLINK1
```

I found that although the size of header calculated as **PRS-HEADZ** above was correctly set before the Linkage Block was linked into the Thing list, it had been altered by the time linkage was complete. I traced the offending code to a subroutine called **th_newth** in the SMSQ/E source code. It resides in the file **util_thg_usage_asm**. This code quite definitely sets the address of the Linkage block into a long word six bytes earlier than the start of that block. As you can see, in my program that is just where the size of header is set in the Thing Header.

Explanation

From the point of view of an application programmer all this might seem extremely odd. But a systems programmer will see it differently. The application programmer should know that when a program is removed any of its channels still open are closed and any space allocated to it is returned to the heap. These useful facts allow application programmers to save code relying on the operating system to clean up after them. They do not need to know how the cleaning up is done.
The systems programmer has to be aware of the details of the cleanup. After all he has to do the coding of this. So how is it done? When a job is removed the operating system must go through the channel table closing all those channels owned by that job. It must also be the case that all allocations from the heap are examined and those owned by the job being removed returned to the heap. Of course, if the allocated area happens to contain a Thing linkage block it will be pretty disastrous if the area is returned to heap without previously being unlinked.
Each area allocated from the common heap has a 16-byte header. I can only find one reference to this header in any of the relevant manuals. That is the Appendix R on page 336 of Dicken's QL Advanced User Guide. However, **keys_chp** in the SMSQ/E source code perhaps shows more clearly what this header contains. There are two types of block, those which are owned by a job and those which are

free space. The contents of the four long words of the header are slightly different in the two cases and appear to be as shown here.

Free Space

Address	Value
$00	Length of area
$04	Relative pointer to next free space
$08	-1 (SMSQ/E), 0 (QDOS)
$0C	0

Allocated space

Address	Value
$00	Length of area
$04	Pointer to driver linkage or 0
$08	ID of owner job
$0C	Address of flag byte set when space released or 0

All device drivers have a long word containing the link to the next driver and, three long words later, the address of the "close" routine. It seems clear that if space allocated to a driver's linkage block is to be returned to the heap it should first be unlinked. This can be done by the 'close' routine. Indeed, when a job is removed the code returning allocated space owned by the program first calls a 'close' routine if there is one.
The Thing linkage block is treated just like a device driver. Thus, when a linkage block is linked into the Thing list, the address of the linkage block is placed six bytes before the start of the block, which is what I was complaining about earlier, and also the address of a 'close' routine is set three long words on from the start of the linkage block. It is this which ensures that the Thing linkage block is unlinked before the space in which it has been living, is discarded.
But all this is on the assumption that the linkage block is the first, or only, item in the allocated space. It is a pity that the manuals do not mention this important fact. I suggest that the wording in the manual should have been that the linkage block 'must be the first or only item in space allocated from the common heap . . '.

Final Comments

First

I did not want to alter **NET_PEEK** so that it would create a Thing linkage block in its very own space. Instead I moved the linkage block down a bit by adding the line

```
PSEUDO_H DC.L      0,0,0,0    ; Pseudo header for Thing linkage
```

just before the Thing linkage block.
This allowed the linking of the linkage block to occur without damaging **NET_PEEK**'s Thing header.
Since I did not allow **NET_PEEK** to be **LRESPR**d from any BASIC but the master, I knew that the owner of the linkage block would never be removed so that the unlinking would never need to occur by space being returned to the heap.

Second

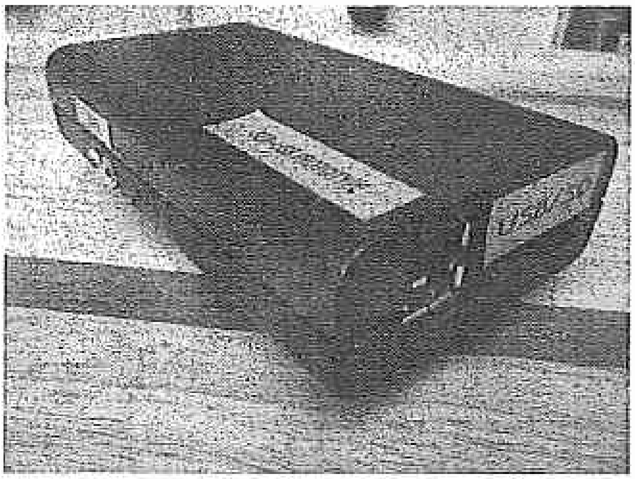
Having altered **NET_PEEK** in the way I have mentioned I loaded it into UQLX and found that it both **LRESPR**d to a Thing and also came to life by `EXEP NET_PEEK` I assume that the very large space erroneously assigned to the header by the Thing linking had made it impossible to set **NET_PEEK** as a program started from the executable Thing called **NET_PEEK**.

Last

This all started because of the cry 'LRESPR doesn't work'. Well, so far that has not been explained. I conjecture that the use of an early ROM in the particular UQLX meant that **LRESPR** failed because some jobs had already been loaded which means that the resident space is no longer available. Of course with SMSQ/E we do not have that problem.

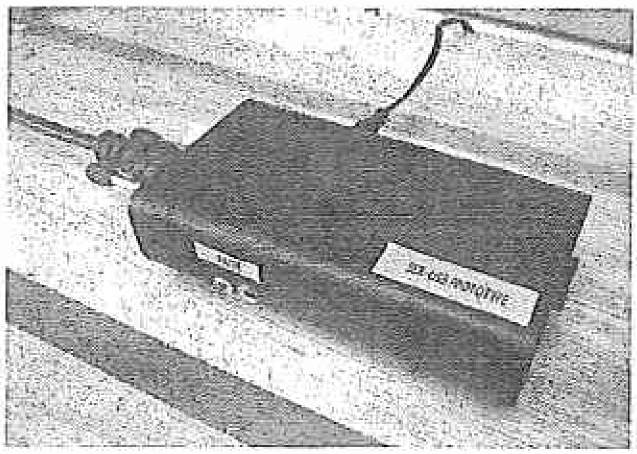
USB Storage for the Sinclair QL and Clones - Ser-USB

by Adrian Ives
Memory Lane Computing Ltd

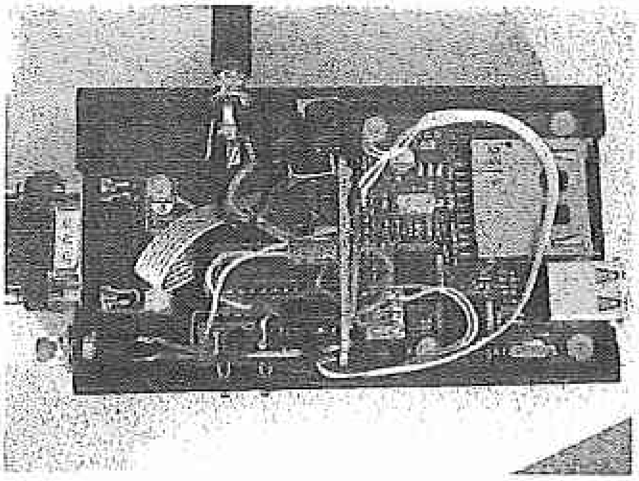


Development on this project started back in November of 2009 after Tony Firshman mentioned the USBWiz on the QL Users list. I purchased one of the units and set about constructing a prototype with a view to investigating the feasibility of writing a driver. Very early on it was clear that the speed of the serial port would be a limiting factor and I had several e-mail discussions with Tony about the use of superHermes as the best solution to increase performance.

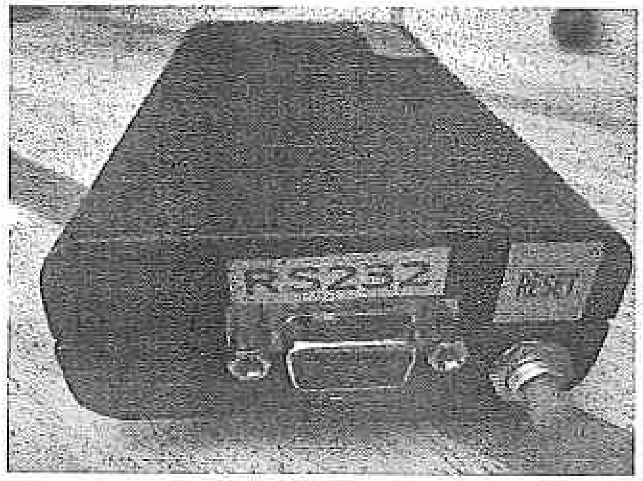
The prototype consists of a black box (12cm x 7cm x 2.5cm) containing a USBWiz module, a MAX232 RS232 to TTL level converter chip (to match the levels on the QL serial port to the TTL inputs on the USBWiz) and some supporting logic to drive a pair of RX/TX status LEDs. Later I added a reset button, although it is rarely needed.



The image below shows what's inside the case. The USBWiz module is on the right hand side. The MAX232 level converter circuit is the small piece of vertically mounted Veroboard in the centre. The 74LS04 and transistors to drive the RX/TX LEDs are the small boards on the bottom left.



At one end of the box is the 9 pin D connector for the RS232 connection.



At the opposite end there is one SD card slot and two USB ports (see next page).

The unit is powered by a small external 5V adapter (not shown here).

Coding began with the standalone File Manager, which was working in late December of 2009. This program made it possible to copy files to and from a FAT format SD card or USB hard drive/memory stick. The QDOS file header is

JOCHEN MERZ SOFTWARE

Kaiser-Wilhelm-Str. 302
47169 Duisburg, Germany

Fax +49 203 502012
EMail: SMSQ@J-M-S.com

SEVERAL PRICE CUTS!

QPC2 Version 3 + SMSQ/E Software QL-Emulator for PC's		**EUR 59,90**
QPC2 Version 3 - Upgrade from QPC2 Version 2		**EUR 19,90**
QPC2 Version 3 - Upgrade from QPC2 Version 1		**EUR 39,90**
QPC Print - printer emulation driver for QPC		**EUR 39,90**
BUNDLE: QPC2 and QPCPrint	**ONLY**	**EUR 79,90**
Agenda Agenda program for WMAN and Prowess	[V1.09]	**EUR 14,90**
Suqcess Database front-end for WMAN	[V2.05]	**EUR 19,90**
QD2003 Pointer-Environment-Editor	[VB.01]	**EUR 29,90**
QD2003 Upgrade from Version 9 and older	[VB.01]	**EUR 14,90**
QMAKE Pointer-driven MAKE for GST/Quanta Assembler	[V4.31]	**EUR 14,90**
BASIC Linker	[V1.21]	**EUR 14,90**
WINED Floppy/Harddisk Sector- & File-Editor	[V1.26]	**EUR 14,90**
FiFi II File-Finder - Extremely useful!	[V4.31]	**EUR 14,90**
FiFi II Upgrade from Fifi Version 3 or older	[V4.31]	**EUR 9,90**
EPROM Manager	[V3.02]	**EUR 14,90**
QSpread2003 Spreadsheet Program	[V4.04]	**EUR 29,90**
QSpread2003 Upgrade from Version 3 and older	[V4.04]	**EUR 14,90**
QPAC I Utility programs	[V1.11]	**EUR 19,90**
QPAC II Files, Jobs & other Things	[V1.45]	**EUR 29,90**
QTYP II Spell checker	[V2.17]	**EUR 19,90**
QPTR Pointer Toolkit	[V0.30]	**EUR 29,90**
DISA Interactive Disassembler	[V3.04]	**EUR 29,90**
CueShell	[V2.14]	**EUR 29,90**
CueShell for QPC	[V2.14]	**EUR 14,90**
SER Mouse software mouse driver for serial mice		**EUR 10,00**
EasyPTR Version 4	[V4]	**EUR 59,90**
EasyPTR Version 4 - Upgrade from earlier versions	[V4]	**EUR 39,90**
QDT - QL Desktop program		**EUR 59,90**
QMENU Version 8 - with new, printed Manual	[V8.02]	**EUR 24,90**
QMENU Version 8 - Update from earlier Versions, also with printed manual		**EUR 17,90**
QMENU Version 8 - New/Update for QL Today subscribers, with prtd manual	**ONLY**	**EUR 14,90**

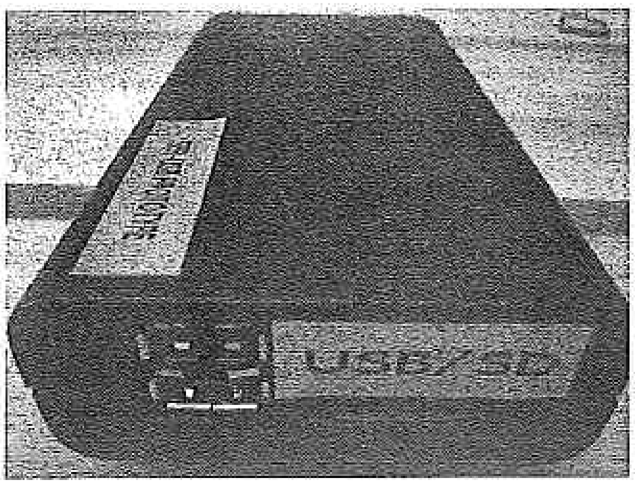
Please add EUR 4,90 for postage to all destinations - Germany, Europe, Wordlwide!

We accept VISA, MasterCard & Diners Club online and offline!
Details for money transfers:

- Deutschland: Jochen Merz, Account 493 50 431, Postbank Essen, BLZ 360 100 43
- Österreich: Jochen Merz, Account 85055317, PSK Wien, BLZ 60000
- Switzerland: Jochen Merz, Account 60-690080-4, PostFinance, Clearing-Nr. 09000
- The Netherlands: Jochen Merz, Gironummer 3258439, Postbank NL Amsterdam
- and from all other countries in EUR with IBAN and BIC to account
  Jochen Merz, Deutsche Postbank AG, IBAN: DE21 3601 0043 0611 1004 37 / BIC: PBNKDEFF 360
- UK customers can pay in £ (convert EUR prices above to £ by multiplying with 0.90) to
  Jochen Merz, Account 83795395, Citibank UK, Sort code 30-00-45
  or send cheques in £ - no fee for UK sterling cheques!
- US customers can pay in US$ (convert EUR prices above to US$
  by multiplying with 1.45) - no fee for US cheques in US$!

Cheques payable to Jochen Merz only!

Price list valid until 30th of June 2011



stored using the same format adopted by Q-emuLator when it saves QDOS files to Windows media (i.e. storing the header as a 64 byte block at the beginning of the file)

I then started development of the driver by examining the sources for the last QUBIDE ROM build with the intention of stripping out the hardware specific code and replacing it with an "ATAPI Emulator" that would fool the rest of the driver into thinking it was talking to a standard IDE hard drive controller.

By late January 2010 the first build of the driver was running but it was unstable. I could not find a way around the problem of supervisor mode code (trap and scheduler services) invoking the serial driver. I could mount the card, load the FAT and do an initial DIR but anything else inevitably resulted in a lock-up.

There then followed a long pause during which my wife and I relocated to Cornwall. Early in 2011 I resolved to dust off the USBWiz project and try one more time to get something working. It was obvious that I had to do something completely different. The ATAPI Emulator approach worked in theory, and almost in practice, but it was messy and I couldn't get around the problem of doing serial I/O inside the driver which led to frequent crashes. Thus the new version of the driver was born. It does all of its I/O in user mode, using a queue to store pending requests which are then picked up by a Queue Manager job. The Queue Manager then spawns a Read or Write job to perform the actual communication with the USBWiz over the serial port.

The driver is capable of doing 100% asynchronous I/O - a read or write request immediately returning after placing a request the I/O queue. In practice, reads have to be made synchronous, otherwise application software using the standard I/O traps would not know that the data had not actually been read yet! The driver handles this by going into a wait state after issuing a read request, checking the queue periodically until the request has been completed before returning. There will be an alternative interface to allow application programs to issue asynchronous read requests if they wish to use this facility. Writes are always asynchronous, data being transferred from a copy of the output buffer created at the time the request is placed in the I/O queue.

The USB device still appears to the system as a normal QDOS device driver, but behind the scenes the way it is doing its I/O is quite different. The new device driver has the name USB; By default USB1 is the SD card slot, USB2 and USB3 refer to the USB ports which can mount standard external hard drives or memory sticks.

Although it sounds simple, QDOS has lots of ways of making this solution anything but! Anyway, on the 1st February 2011, my version 0.02 prototype driver successfully wrote a plain text file of 1.7K to a native QDOS formatted SD Card. After a restart (to ensure that the slave block cache was empty and the data really was coming off the card) the driver successfully read it back again.

```
USB1
21076/21168 sectors
x_ref_txt

alloc_asm▒balloc, falloc, find_slot▒
atapi_asm▒chs_to_lba, read_lba, write_lba▒
basics_asm▒basic_procs▒
block_asm▒prepare_blockcall, r_blk0, r_pblk, r_pblk0, read_t
boot_asm▒boot_init▒
close_asm▒do_close▒
core_asm▒calc_csum, check_csum, dotidy_map, drv_inst, drv_ir
dbgout_asm▒dbg_out, dbg_out2w, dbg_outb, dbg_outl▒
delete_asm▒chk_empty, del_map, delete▒
extras_asm▒fil_ver, med_inf, med_xinf, q_date▒
file_asm▒in_load, out_save▒
flush_asm▒flush, flush_slaves▒
forced_asm▒do_forced, get_addr, get_slaved, getiblk, slaved▒
format_asm▒do_format▒
intserv_asm▒sched, driver_fifty_hz▒
io_asm▒▒do_IO, io_exit▒
iod_asm▒▒get_1sector, iod_tbl▒
ioru_asm▒chk_pend, in_byte, in_line, in_pend, in_str, out_by
libcode_asm▒LongDivS, LongDivU, OutLong, q_str, RetInt, RetL
makedir_asm▒mk_dirs▒
open_asm▒complete, do_open, exit_open▒
posit_asm▒pos_abs, pos_rel, rd_head, set_head▒
rdsect_asm▒drive_read,drive_read_async▒
rename_asm▒rename, trunc▒
setup_asm▒base, hard_add▒
text_asm▒atapi_msg, bad_fat, bad_msg, blksz_msg, block_msg,
▒ide_msg, load_fat, mem_msg, mstr_err, nold_msg, qdos_err,
trashcan_asm▒dotrash▒
usbdev_asm▒boot, init▒
usbdrv_asm▒drive_capacity, drive_cmd, drive_nbusy, drive_sel
▒usbwiz_chkpipe, usbwiz_init, usbwiz_link, usbwiz_queue_adc
▒usbwiz_spawn_qm▒
util_asm▒dir_search, pre_conts▒
```

```
[illegible] 'USB_KRN_bin'
Serial USB driver [illegible] (2011)
[illegible] to a USBWiz [illegible] 9600 baud
Loading and checking FAT ...
Dir usb1_
[illegible]
```

It had been a very long journey but, finally, I was in business!

There is still a lot of work ahead. For a start the FORMAT routine doesn't work yet; formatting is currently achieved by running an S*BASIC utility. There are issues to be solved around the I/O queue filling up when memory is limited, and the whole driver needs optimising to strip out a lot of residual code from the original QUBIDE driver that is now completely redundant. Finally, it will all need testing to destruction in an environment with other device drivers running, to identify and resolve any conflicts and performance issues. I do not underestimate the amount of work still ahead!

On the plus side, the current version supports driver-level access through a pipe, allowing commands and enquiries to be sent to it in real time. This will ultimately be thing-based. The driver also has an S*BASIC interface with commands to get/set the baud rate, read/write sectors and generally configure the driver.

To get a flavour of the S*BASIC interface here is the current list of procedures and functions that the driver installs:

Procedures

USB_USE: Change the device name of the driver
USB_DRIVE: Link/Unlink drives and partitions
USB_CTRL: Change driver settings for a drive
USB_GETSEC: Read a single sector
USB_PUTSEC: Write a single sector
USB_SETBAUD: Set the baud rate for USBWiz communications
USB_AUTO_FLUSHMAPS: Enable/Disable automatic map updates
USB_FLUSHMAPS: Force the maps to be written from memory (if automatic updates are disabled)
USB_DEBUG: Set the debug flag to the supplied byte value

Functions

USB_CTRL%: Return the current CTRL setting for a drive
USB_DEVB: Return the base of the driver's variables
USB_BAUD: Return the current USBWiz baud rate
USB_PIPE_W: Return the Channel ID of the inbound driver command pipe
USB_PIPE_R: Return the Channel ID of the outbound driver command pipe

I will also be producing a pointer environment version of the standalone File Manager (whilst keeping a stripped down character mode version for use on unexpanded systems).

Development to date has been carried out using QPC2 running on a Windows 7 64 bit laptop, an Aurora Super Gold Card QL with superHermes, and a bog-standard unexpanded QL. The QPC environment has been invaluable for doing high speed edit/assemble cycles, enabling me to combine the best windows tools with those for the QL in my development environment.

It is my intention to make the base driver (once completed) open source. Add-on software such as file managers and extensions will be commercial products, as will the Ser-USB hardware itself.

Adrian adds:

Build 009 of the driver was completed and tested today (6th February 2011). This fully implements the io.formt trap, meaning that you can now issue the command:

```
FORMAT USB1_
```

Note: The Ser-USB format routine is much simpler than the QUBIDE original; it only prompts the user for a total size in Megabytes and the block size and then does the rest for you. If the combination of total size and block size would result in a partition with more than 65536 blocks, the user is advised and asked to select a larger block size and/or smaller partition size.

Formatting is also a background process, meaning that in an S*BASIC session the FORMAT command completes almost immediately, leaving the drive formatting in the background. As Ser-USB doesn't do low-level formatting the process is as quick as the time it takes to write a new Map and root directory.

Editor's comment: what an interesting product - missing for a very long time! Best wishes to Adrian to turn it soon into a commercial product - we are sure many users are waiting for it!

XCHANGE – Part 2

by David Denham

I hope you have managed to set up your copy of Xchange by now. If you have got round to reading some of the documents you might have noticed that one of the things which has changed quite substantially for the better is the printer driver handling.

Not only is there more flexibility in where you can store the printer drivers - Xchange is able to handle sub-directories for example - it is possible to have more than one program running with different printer drivers should you need to do so, and the programs can look in different places for them as well as the printer driver themselves being able to have different names if you wish (you don't have to stick to just printer_dat as you would with the original programs). Printer drivers can now have up to 50 translate sequences, many more than the original QL versions. But this added flexibility can be daunting at first as all the possible permutations can be difficult to grasp.

If you are used to the original Quill, Archive and Abacus, you will be familiar with the process of using the install_bas program. It was quite simple - you run a program called install_bas to edit and choose a printer driver (all of which it stores in a single file called install_dat), and the particular printer driver selected and installed for these programs was called printer_dat, a file which could be used by all three of the programs. Easel is a little bit different, it uses a printer driver called gprint_prt to print screen dumps - we'll discuss Easel separately.

PEDIT

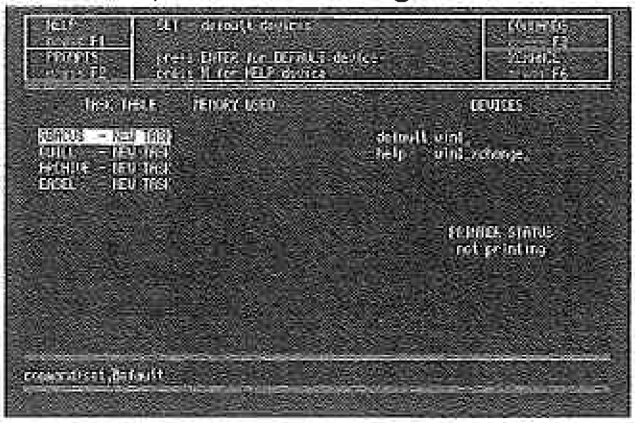
Printer drivers for Xchange are edited with a program called PEDIT and there are two versions of this program. One is a superbasic program called pedit_bas and the other is a compiled basic program called pedit. You LRUN the former and EXEC the latter.

The information for the Xchange package's printer drivers is stored in a file called pedit_dat. Normally, when you choose a driver, it is created as a file called xchange_dat, although there are one or two other possibilities - Xchange can use the older printer_dat files too, as well as drivers named after the program which is to use them.

Before we install our first printer driver, we need to know how and where the Xchange programs look for these printer driver files, so we can plan where to put them.

When printing, the Xchange programs will look for the printer driver on either the help file drive or the default drive. When Xchange starts you can set these by using the SET command - press F3 then S for Set. If you now press H it will ask you for the name of the help drive - it will offer the current setting (which might be win1_xchange_ for example) and you can type in a different drive and directory name. If instead of pressing H, you pressed ENTER to set the default device, you would be setting the drive and directory name where Xchange would save files to if you omitted the drive name in the filename.

This might be a bit easier with the aid of a screen dump. Have a look at figure 1.



```
SET - default devices
press ENTER for DEFAULT device
press H for HELP device

TASK TABLE      MEMORY USED                    DEVICES

ABACUS  - NEW TASK                  default win1_
QUILL   - NEW TASK                  help    win1_xchange_
ARCHIVE - NEW TASK
EASEL   - NEW TASK

                                          PRINTER STATUS
                                          not printing

command>set,default
```

Fig.1: Using the SET command to set default and help drives

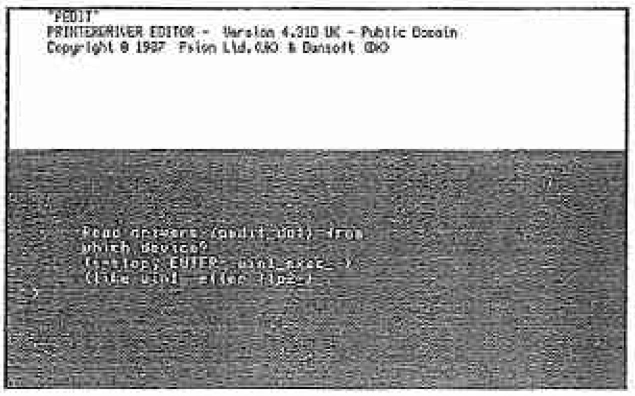
You can see from figure 1 that we have selected the command Set from the commands menu in Xchange's startup screen. The prompts area at the top tells you what to do if you are unsure. Towards the top right, it shows the DEVICES settings - under the word DEVICES you can see what the default and help drives are currently set to. Now follow what it says at the top to type in a change of either of those settings at the bottom, where it says 'command›set,'.

If, like me, you store everything in win1_xchange_ and you are not too sure what to set these to, make them all win1_xchange_ if you wish. Just remember that the default drive is for saving files with, while the help drive is for when it needs to look for its help files and the printer drivers. Actually, if it doesn't find printer drivers in one place, it will usually look in the other, just to confuse you!

Going back to the PEDIT program, we'll look at how to select and create a printer driver using PEDIT. First, start the program with an EXEC or EXEC_W command:

```
EXEC win1_xchange_pedit
```

This should start with a display like the one shown in figure 2.



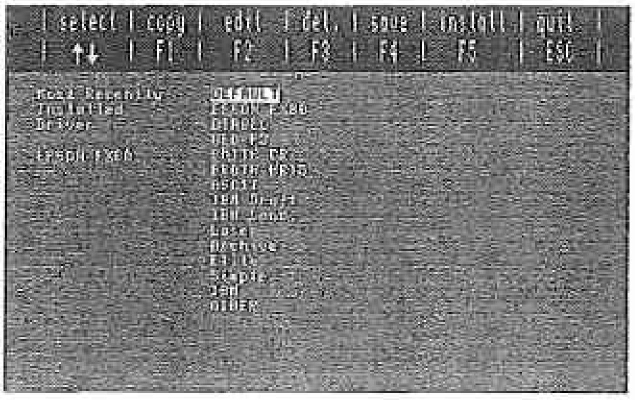
```
"PEDIT"
PRINTERDRIVER EDITOR - Version 4.31D UK - Public Domain
Copyright © 1987 Psion Ltd.(UK) & Dansoft (DK)

Read drivers (pedit_dat) from
which device?
(default: ENTER: win1_xchange_)
(like win1_ , flp1_ , flp2_)
>
```

Fig.2: The pedit program starting up

The first thing it asks you to enter is where are the collection of printer driver files (a file called pedit_dat) stored. At this point you can enter s to stop the program, just press ENTER to accept the default shown (which is usually the same as your PROG_USE default setting in superbasic. Alternatively, you can enter a drive name like FLP1_ or win1_xchange_ , wherever you have stored pedit_dat.

The program now loads the information in pedit_dat and then asks what type of printer port you will be using. You can press ENTER to use a serial port SER1 or SER2, or you can press the space bar to specify a parallel port (PAR) or some other non standard device name.

You now get a quick list of the printer drivers held in pedit_dat, then it goes to the screen shown in figure 3.



```
| select | copy | edit | del. | save | install | quit |
|   ↑↓   |  F1  |  F2  |  F3  |  F4  |   F5    |  ESC |

Most Recently        DEFAULT
Installed            EPSON FX80
Driver               DIABLO
                     NEC P2
EPSON FX80           [illegible]
                     [illegible]
                     ASCII
                     IBM Draft
                     IBM Cont.
                     Laser
                     Archive
                     [illegible]
                     Simple
                     Jeni
                     OTHER
```

Fig.3: The main pedit screen

This looks pretty similar to how the printer editing was done in the original Psion programs, so this will be pretty intuitive.

If you want to edit one of the drivers shown, use the arrow keys to move the highlight to that one, then use the function keys shown at the top - F2 to edit the driver, for example.

Editing is done in pretty much the same way as with the original Psion programs. Move the highlight down to the feature you want to change with the up or down arrow keys, press the left or right arrow keys to change something and change it exactly as you have done with the older Psion programs.

There are two new features to consider at this point. You can now have up to 50 translate sequences, but do bear in mind that the total length of the printer driver has to be no more than 256 bytes long, so 50 very long translate sequences might make it run out of space. You can choose whether to have a small number of long sequences, or up to 50 fairly short sequences of control codes in there.

When you have finished editing the driver, you can press F4 to save it and F5 to install it as the printer driver for Quill, Abacus and Archive. The same one can be used for all three if you wish.

After pressing F5, it asks you where to save this printer driver. This will usually be where your main Xchange files are stored, in my case win1_xchange_. If you just press ENTER it will save it to the same place as you loaded it.

Next, it asks 'For which program ?'

At this point I need to discuss another new feature. You can have one printer driver to serve all programs, called xchange_dat.

Alternatively, you can specify it's to be used with Abacus, Archive, or Quill - it will save it with a different name to match that particular program. It asks:

For which program?
0: Xchange 1: Abacus 2: Archive 3: Quill 4: Other

So if you enter 0, it calls it xchange_dat when it saves it. If you enter 1 it gives the printer driver the name abba_dat. Enter 2 and it saves it with the name archv_dat. Enter 3 and it will give it the name quil_dat. Enter 4 and you can type in a short name. If you enter the word test, the printer driver will be saved as a driver called test_dat. This last option lets you create a special printer driver with a special name. If you opted to call the drivers abba, archv or quil (options 1 to 3), these are referred to as the 'generic names'.

The purpose of this might not be immediately apparent.

The explanation lies in the new flexible printer driver naming and searching. To understand how it works, we need to know how the programs look for their printer drivers.

Step 1 - they look for the printer driver on either the help drive or, if not found there, on the default drive.

Step 2 - it decides on the name of driver to look for, depending on what it finds. This can be one of three names.

1. It looks for a printer driver with the same name as the JOB name of the task we started in Xchange. Say we started a copy of Quill, giving it the name 'work'. Quill would look what its own name is (in this case, work) so it would check to see if it could find a printer driver called work_dat and if found, it uses that one.
2. If that was not found, it looks for printer drivers called quil_dat, abba_dat or archv_dat as appropriate - what I referred to as Generic names above.
3. Finally, it tries to find xchange_dat if neither of the other options was found.

Actually, there is a fourth option - it can use the old printer_dat from the original Psion packages. There is little point in doing this as the new Xchange ones are more flexible.

So, you can see there is a search sequence involved in deciding which driver to use. While this is quite likely to be confusing at first, once you realise that, for example, you have two copies of Quill going, for example, or an Abacus and a Quill, with careful choice of job names and printer driver names, each program running can have its own printer driver should you need to do so.

So if I have one copy of Quill running for work documents, I could give that a printer driver which sets the printer to high quality printing. And if the other Quill was for home use where high quality printing was not required, it could have a separate printer driver which sets draft quality printing to save ink.

This is why, if you wish to make use of this flexibility you need to try to understand how the programs find and decide which driver to use, and where they load them from.

- They can be loaded from either of two drives (default or help drives)
- Which driver name to be used, the name search sequence I referred to above.

So, since the driver can be in either of two drives, and three possible names, there can be up to six possible filenames which the program concerned might use. If you are unable to get your head around all this, or you find that one driver is suitable for all your needs, just call the driver xchange_dat and all will be well.

Once you have finished installing the drivers, press ESC to quit from the pedit program.

EASEL

Printing from Easel works in pretty much the same way as from the original Easel package. The printer driver is still called gprint_prt and is loaded from the help device set for Xchange.

If you wish to change printer driver for Easel you can do so with the Print command within Easel.

Press F3, then P for Print, and it shows the options for Print, Epson plotter, HP Plotter, Screen dump to file and I to install a new graphics printer. Then, just follow the prompts. A small selection of Easel drivers is supplied - they have names ending with _srt for serial printers, or _prt for parallel printers.

A new facility is the option to save to a file. You can now choose between three different types of file, depending on what filename extension you use.

1. If you want a standard 32KB QL screen file just give the filename an extension of "_pic" or "_scr". On a standard original QL with the screen at the original address, these pictures can be loaded back in with the command lbytes filename_scr,131072 (or _pic instead of _scr if you used that). Note that if your QL or emulator has a high resolution screen or the screen is not at the same address as on an original QL, this may not work correctly, of course.
2. If you are saving a picture to be used in the QDesign program, save it with a filename which ends with "_cut", e.g. ram1_test_cut. QDesign is an old graphics design program which you don't see much these days.
3. To save a file which is compatible with QPTR applications, you can use any other filename extension. These files have a ten byte header in the file, which carries information about the width and height in pixels, the line width (in bytes) and the screen mode number. These can only be loaded into programs which understand the Qptr file format - a lot of modern graphical and pointer driven programs do.

HINT

The printer drivers for Quill, Abacus and Archive can have quite long code sequences, as long as you don't make them longer than the total limit of 256 bytes I mentioned above. This allows you, with a bit of care and forethought, to generate some rather special printer drivers for specialised applications. One example I have seen (you can get this one from Dilwyn Jones's website at **http://dilwyn.me.uk/psions/index.html**) is one which

QUANTA

Independent QL Users Group

World-wide Membership is by subscription only,
Offering the following benefits:
Bimonthly Magazine - up to 52 pages
Massive Software Library - All Free!
Free Helpline and Workshops
Regional Sub-Groups. One near you?
Advice on Software and Hardware problems
Subscription just £14 for Full Membership

PayPal (see QUANTA Web Site),
Cash, Cheques and Postal Orders Accepted

Now in our Twenty Eighth Year

Further details from the Membership Secretary

John Gilpin, 181, Urmston Lane,
Stretford, Manchester, M32 9EH (UK).
Tel. +44 (0) 161 865 2872
Email: membership@quanta.org.uk

Visit the QUANTA Web Site

Next QUANTA Sponsored Event

Annual General Meeting 2011 and Workshop
Date: Saturday/Sunday 16th/17th April 2011
Workshop from 12.00 Noon (Doors open 10 am for setting up) to 5.00 pm Saturday
And 9.00 am to 1.30 pm Sunday
Annual General Meeting 2.00 pm Prompt Sunday.

Venue: 3rd Davyhulme Scout Headquarters "The Endeavour", Conway Road, off Lostock Road, Davyhulme, Manchester. M41 0TF. Near M60 J9.

Full details from Chairman@quanta.org.uk lets you print text output from Quill as an html file (a web page) instead of a paper printout. What it seems to do is set the preamble codes to generate the html and header codes ‹html› and ‹head› instead of printer control codes, and likewise using the postamble codes to send the ‹/html› to finish the page. It also seems to use the bold, high and low codes to generate specific types of html such as bold codes. A bit specialised, but if you want to put together a very simple web page, using Xchange Quill, it's good enough for the most basic of designs. Another driver allows for text to be sent to a file with QL to PC character code conversions. Another one is set up to allow Quill to send its output as plain text to a file.

SUMMARY

The use of printer drivers with Xchange can be as simple as just using one printer driver called xchange_dat. Users with more complex requirements can set up more complex and more versatile printer driver sets. It is a bit much to take in all in one go when you are using Xchange for the first time, but persevere and keep these notes to hand and you'll gradually get used to the printer driver enhancements in Xchange.

Programming in Assembler Part 27 – The Return of WMAN

by Norman Dunbar

Introduction

Before my recent delving into George's SETW, his Easy PEasy utilities and my brief foray into pdf magazine production, I was walking through the various requirements of setting up a window definition using WMAN. This edition continues from where we left off but incorporates George's utilities into the examples. As George has made it easy to write Pointer Environment programs, I think we should make use of his hard work. Code reuse is all the rage!
However, before we start coding, we better take a moment to discuss Application Sub-Windows.

Application Sub-Windows

There are a number of uses for application sub-windows, for the program's display or to hold a menu, correctly known as an *application window menu*. An application window is a variable thing and as such, can be defined in a number of ways. Because of this, and because an application can have multiple application sub-windows, when defining our main window (or the variable parts of our main window) we don't have a pointer to an application window. Instead, we have a pointer to a list of pointers and each of these, in turn, points to an application window definition. The list is terminated by a zero word.

The following is the relevant extracts from a definition of a main window which contains information windows, loose items and a pair of application sub-windows.

```
; Main window definition :
            ...
            dc.w  info_list-*          ; Pointer to information window list
            dc.w  loos_list-*          ; Pointer to list of loose items
            dc.w  appw_list-*          ; Pointer to list of applications windows
            ...

; Application window list :
appw_list   dc.w  appw_0-*             ; Pointer to first sub-window definition
            dc.w  appw_1-*             ; Pointer to second sub-window definition
            dc.w  0                    ; No more application windows

; Application window 0 :
appw_0      ...                        ; Start of definition. See text for details
```

```
; Application window 1 :
appw_1     ...                      ; Start of definition. See text for details
```

The definition of an application sub-window is described below.

```
; Application sub-window definition :
           dc.w  192                ; Width in pixels (+ scaling)
           dc.w  119                ; Height in pixels (+ scaling)
           dc.w    4                ; X origin relative to 0 in main window
           dc.w   18                ; Y origin relative to 0 in main window
           dc.b  256                ; Clear flag. Bit 7 set = clear window
           dc.b    0                ; Shadow depth - must be zero!
           dc.w    1                ; Border width
           dc.w    0                ; Border colour
           dc.w    7                ; Paper colour
           dc.w    0                ; Pointer to pointer sprite, or zero for
arrow
           dc.w    0                ; Pointer to user defined setup routine, or
zero
           dc.w    0                ; Pointer to user defined drawing routine, or
zero
           dc.w    ahit0-*          ; Pointer to hit routine
           dc.w    0                ; Pointer to sub-window control routine, or
zero
           dc.w    0                ; Max allowed X control sections (splits)
           dc.w    0                ; Max allowed Y control sections (splits)
           dc.b    9                ; Selection key - moves pointer into this
window
           dc.b    0                ; Spare byte - must be zero
```

For our current needs, this definition allows us to have a simple application sub-window with no pan and scroll control bars and no menu. The user defined setup and drawing routines are most often defined as zero words to allow WMAN to do the hard work.

Application Sub-Window Hit Routines

The hit routine for an application sub-window is called from within the wm_rptr call either when you HIT in the window, or when you press the activation key for that sub-window. Similar to loose item action routines previously discussed, if the code exits with D0 set to zero, the wm_rptr call will resume again - in other words, control will not return to your own code just yet - unless the hit code sets any event bits in the event vector. This is slightly different from loose item action routines in this respect.

On entry to a sub-window hit routine various registers are set with specific parameters:

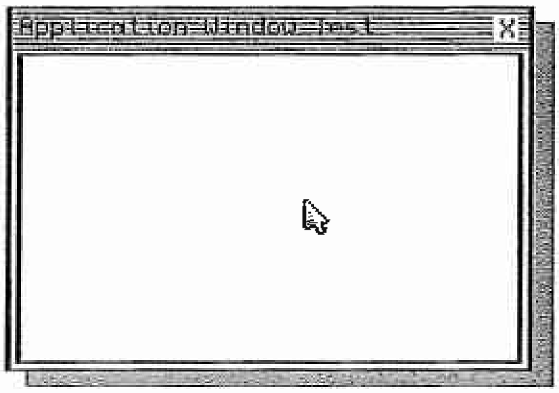
Register	*Description*
D1.L	High word = pointer X position. Low word = pointer Y position in *absolute* screen co-ordinates. Ie, the pointer position within the entire screen and *not* within the program's window or the application sub-window itself.
D2.W	Selection keystroke letter, in its upper cased format, or 1 = Hit/SPACE or 2 = DO/ENTER. If D2 is -1, then the application sub-window was "hit" by an external keystroke.
D4.B	An *event number*. This can only be 0, pt__do (16) or pt__cancel (17) as all other events are handled by WMAN. If you have a loose item with ESC as the selection keystroke, then the loose item action routine will catch the ESC keystroke - the application sub-window hit routine will not see it if the ESC causes the program to exit.
A0.L	Channel id.

A1.L	Pointer to the status area.
A2.L	WMAN vector.
A3.L	Pointer to sub-window definition.
A4.L	Pointer to window working definition.

Hit routines should exit with D1, D5 - D7, A0 and A4 preserved to the same value that they had on entry to the routine. D2, D4, A1 - A3, A5 and A6 are undefined on exit (which means that they don't care what value they have.) The hit code must set the SR according to the value in D0 on exit.
D3, on return from a hit routine, should normally be returned as per its value on entry. It is not used by wm_rptr however, it is used by wm_rptrt (read pointer with return on timeout) from WMAN 1.5 onwards. Wm_rptr ignores the upper word of D3. If your read pointer loop is using the wm_rptrt vector instead, and you have changed the value of D3 within the hit code, you must clear the high word on exit.
It is important to note that WMAN *doesn't* set the event bits for you, it is up to the hit code to do that for you. For example, if someone HITs the application window then the hit routine will be called with D2 = 1 which is also the case also when someone DOs the application window but the pt__do bit in the window byte of the event vector will *not* be set.
On exit, if D0 is clear and the status (Z) bit is set, control will return to the wm_rptr loop and not to your application's code. To return to your own code, the hit routine needs to set at least one event bit in the event vector.
If an error is detected within the hit code, then it should exit with the appropriate error code in D0 and the status register set accordingly.

Example Application Window

As before, we now create a useful (!) demonstration program to show us the simplest use of an application sub window. The program will look like the following when completed and running:



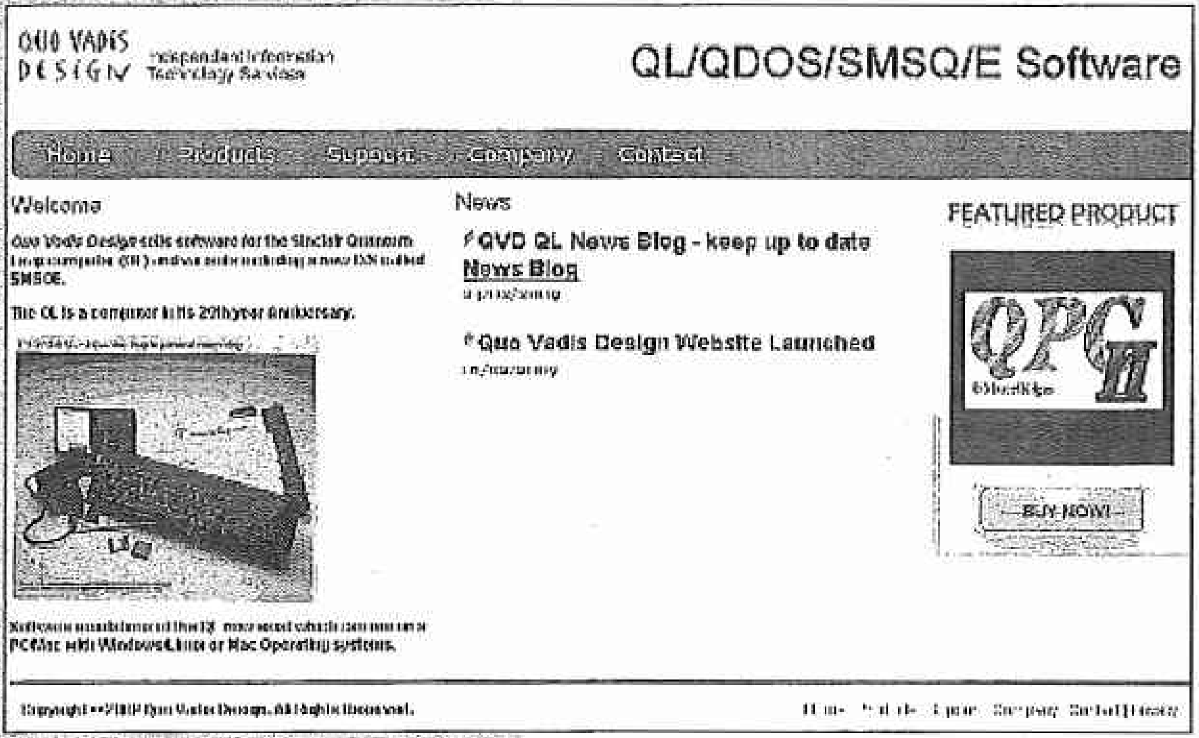
You can see how I'm sticking to accepted QDOSMSQ design standards here can't you!

The window above consists of the following:

- An outline with white paper, a black single pixel border and a shadow. The default arrow sprite is used for the entire window.
- A 'caption bar' consisting of a single information window with green/white striped paper (paper colour 92).
- Within the information window is a single information text object which is simply the program name.
- Also located within the information window is one loose item containing a text object ('X') and this has the ESC keypress code set up to close the window.
- The remainder of the outline is filled with an application window, with white paper and a black single pixel border. (No shadow - they are forbidden for application windows!). This window also uses the default arrow sprite and has a selection key of TAB. This means that if you press the TAB key, the pointer will jump into the application sub-window.

QUO VADIS DESIGN Independant Information Technology Services

www.ql-qvd.com


QUO VADIS DESIGN Independant Information Technology Services

QL/QDOS/SMSQ/E Software

Home · Products · Support · Company · Contact

Welcome

Quo Vadis Design sells software for the Sinclair Quantum Leap computer (QL) and was sold including a new OS called SMSQE.

The QL is a computer in its 25th year anniversary.

Software available for the QL [illegible] PC/Mac with Windows/Linux or Mac Operating systems.

News

QVD QL News Blog - keep up to date
News Blog
[illegible]

Quo Vadis Design Website Launched
[illegible]

FEATURED PRODUCT

QPC II

BUY NOW

Copyright © 2009 Quo Vadis Design. All Rights Reserved.

Home · Products · Support · Company · Contact | Privacy

QL/QDOS/SMSQ/E Software

Check the QL News Blog on our website for updates.

www.ql-qvd.com/blog

Bruce@ql-qvd.com

Quo Vadis Design
38 Derham Gardens
Upminster
RM14 3HA
UK

Tel: +44 (0)20 71930539
Fax: +44 (0)870 0568755

QL Today The QL magazine for all QDOS, QL, SMSQ ... users!

Subscriptions taken online

The window was set up using SETW as follows:

1. When prompted for "name$" enter 'ApplTestWin' (without the quotes) then ENTER.
2. On the "Alter Text" screen.
   - Press N for new, type 'X' (without the quotes) then ENTER.
   - Press N for new, type 'Application Window Test' (without the quotes) then ENTER
   - Press ESC.
3. On the "Alter Sprite" screen, press ESC.
4. On the "Alter Blob" screen, press ESC.
5. On the "Alter Patt" screen, press ESC.
6. Number of main windows = 1
7. Number of Loose Items = 1
8. Number of Information windows = 1
9. Number of IW Objects = 1
10. Number of application windows = 1
11. Application windows menu items = 0
12. For main window 1:
    - Shadow = 2
    - Border size = 1
    - Border colour = colour_ql -› black
    - Paper colour - colour_ql -› white
    - Sprite = arrow
13. Loose Items:
    - Press N for "system palette defaults"
    - Confirm N when prompted again for defaults
    - Border size = 1
    - Border colour = colour_ql -› black
    - Unavailable background = colour_ql -› white
    - Unavailable Ink = colour_ql -› grey
    - Available background = colour_ql -› white
    - Available Ink = colour_ql -› black
    - Selected background = colour_ql -› green
    - Selected Ink = colour_ql -› black
14. Loose Item 1:
    - Type = text
    - Object -› select the 'X' text object
    - Selection key = ESC
15. Information Window 1:
    - Border size = 0
    - Paper = colour_ql -› No 92
16. Object 1:
    - Type = text
    - Object -› select the 'Application Window Test' text object.
17. Application Window 1:
    - Colour = colour_ql -› white
    - Xcsize = 0
    - Ycsize = 0
    - Border size = 1
    - Border colour = colour_ql -› black
    - Paper colour = colour_ql -› white
    - Sprite = arrow
    - Selection key = TAB

18. Main window size: (Use the arrow keys to change the size, press ENTER when correct)
    - Width = 200
    - Height = 140
    - Do you want a variable window = N
    - Set the origin to 0,0 (Press ENTER when correct)
19. Loose Item 1: (Toggle hit/position with F2. Press ENTER when correct)
    - Hit size = 10 x 10
    - Position = 183 x 3
20. Information Window 1: (Toggle size/position with F2. Press ENTER when correct)
    - Size = 200 x 16
    - Position = 0 x 0
    - Object position = 2 x 2
21. Application Window 1: (Toggle size/position with F2. Press ENTER when correct)
    - Size = 192 x 119
    - Position = 4 x 18

When you have completed this procedure, and SETW has exited, you should save the file 'ram1_ApplTestWin_asm' to a safer place. The file should look like the following, although I have added some extra comments to my copy of the generated code.

```
; ApplTestWin_asm

; Undefined Labels - need to be defined elsewhere in my own code.
;       ahit0   - application window 0 hit action routine.
;       afun0_0 - Loose item 0 hit action routine.

; Labels for External Use
;       wst0    - Window status area
;       wd0     - Window definition address
;       ww0_0   - Window default size
;       ww0_1   - Window button size

SYS_SPR  dc.w   0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25
         dc.w   26,27,28,29,30,31,32,33,34,35,36,37

; Definition of all text objects here

txt0     dc.w       txt0_e-2-txt0
         dc.b       "X"
txt0_e   ds.b       0
         ds.w       0

txt1     dc.w       txt1_e-2-txt1
         dc.b       "Application Window Test"
txt1_e   ds.b       0
         ds.w       0

; Application window list.
app_list0
         dc.w       appw0-*
         dc.w       0

; Application windows 0 definition.
appw0    dc.w       192       xsize
         dc.w       119       ysize
         dc.w       4         xorg
         dc.w       18        yorg
         dc.w       256       flag
         dc.w       1         borw
         dc.w       0         borc
         dc.w       7         papr
         dc.w       0         pspr *
         dc.w       0         setr *
```

```
        dc.w        0          draw *
        dc.w        ahit0-*    hit *
        dc.w        0          cntrl *
        dc.w        0          nxsc
        dc.w        0          nysc
        dc.b        9          skey
        dc.b        0          spr1

; Information Object(s)
pobl0   dc.w        138        xsize
        dc.w        10         ysize
        dc.w        2          xorg
        dc.w        2          yorg
        dc.b        0          type
        dc.b        0          spar
        dc.l        0          Ink, xcsize,ycsize
        dc.w        txt1-*     pobj *
        dc.w        -1

; Information window(s)
infw0   dc.w        200        xsize
        dc.w        16         ysize
        dc.w        0          xorg
        dc.w        0          yorg
        dc.w        0          flag
        dc.w        0          borw
        dc.w        526        borc
        dc.w        92         papr
        dc.w        pobl0-*    pobl *
        dc.w        -1         end

; Loose item(s)
litm0   dc.w        10,10      xsize, ysize
        dc.w        186,3      xorg, yorg
        dc.b        0,0        xjst, yjst
        dc.b        0,3        type, skey
        dc.w        txt0-*     pobj *
        dc.w        0          item
        dc.w        afun0_0-*  pact *
        dc.w        -1         end

litm1   dc.w        16404,12   xsize, ysize
        dc.w        0,0        xorg, yorg
        dc.b        0,0        xjst, yjst
        dc.b        0,0        type, skey
        dc.w        0          pobj *
        dc.w        0          item
        dc.w        0          pact *
        dc.w        -1         end

; Window definition
wd0     dc.w        200        xsize
        dc.w        140        ysize
        dc.w        0          xorg
        dc.w        0          yorg
        dc.w        258        flag
        dc.w        1          borw
        dc.w        0          borc
        dc.w        7          papr
        dc.w        0          sprt *
        dc.w        1          curw
        dc.w        0          curc
        dc.w        7          uback
        dc.w        255        uink
        dc.w        0          ublob *
        dc.w        0          upatt *
        dc.w        7          aback
```

```
          dc.w      0         aink
          dc.w      0         ablob *
          dc.w      0         apatt *
          dc.w      4         sback
          dc.w      0         sink
          dc.w      0         sblob *
          dc.w      0         spatt *
          dc.w      0         help
          dc.w      200       xsize
          dc.w      140       ysize
          dc.w      infw0-*   pinfo *
          dc.w      litm0-*   plitem *
          dc.w      app_list0-* pappl *
          dc.w      16384     xsize
          dc.w      12        ysize
          dc.w      0         pinfo *
          dc.w      litm1-*   plitem *
          dc.w      0         pappl *
          dc.w      -1

; Sizes
ww0_0     equ       290
ww0_1     equ       148

; Status Areas
wst0
          ds.b      65
wst0_e    ds.b      0
          ds.w      0
```

Example Program

Having defined our application window test definition, we need a program to run it. However, we must also decide what the program is intended to do when running. As this is our first program with an application window, we will simply write some information to the application window when 'things' happen.

I mentioned 'code reuse' above, so the following is based very heavily on George's example code, with (I hope) all the unnecessary bits removed. Unnecessary, that is, for this example of mine!

The following is enough of a test harness to get our newly designed window up and running, but only the ESC key and the 'X' loose item works. The rest of the program will come later.

In the source code that follows, where I use the 'in' or the 'lib' commands, you will need to replace 'win1_georgegwilt_' with the location of the files being included. Unless you have exactly the same source setup as I do!

We start, as ever, with a standard QDOSMSQ job header and then pull in the various include files from Easy PEasy. Three offsets into the job's data area are then defined.

```
          bra.s start
          dc.l  0
          dc.w  $4afb

fname     dc.w  fname_e-fname-2
          dc.b  "Application Window Test 1"
fname_e   ds.b  0
          ds.w  0

; We need the various equates files etc.

          in win1_georgegwilt_peass_keys_pe
          in win1_georgegwilt_peass_qdos_pt
          in win1_georgegwilt_peass_keys_wwork
          in win1_georgegwilt_peass_keys_wstatus
          in win1_georgegwilt_peass_keys_wman
          in win1_georgegwilt_peass_keys_wdef
```

```
id      equ 0                   ; Offset to channel id storage
wmvec   equ 4                   ; Offset to WMAN vector storage
slimit  equ 8                   ; Offset to storage for return from IOP_FLIM.
;                                 The storage is X-size, Y-size, X-org, Y-org - all words.
```

Following on from the above, we have the job's start and initialisation code. As the vast majority of this has been explained before in the introductory article on George's Easy PEasy, I shall not go into it again here. See *Easy PEasy Part 1 in Volume 14 Issue 3* for full details.

```
start   lea (a6,a4.l),a6        ; Make A6 point to the job's dataspace
        bsr op_con              ; Open a con channel
        move.l a0,id(a6)        ; And store the channel id
        moveq #iop_pinf,d0      ; Trap to get Pointer Information
        moveq #-1,d3            ; Timeout
        trap #3                 ; Do it
        tst.l d0                ; Is ptr_gen present?
        bne sui                 ; No, bale out via SUI
        move.l a1,wmvec(a6)     ; Yes, store the WMAN vector
        beq sui                 ; Oops! WMAN wasn't actually found

flim    movea.l a1,a2           ; The WMAN vector is required in A2
;                                 The channel id is already in A0
        lea slimit(a6),a1       ; Result buffer
        moveq #iop_flim,d0      ; Query maximum size of window
        moveq #0,d2             ; D2 is required to be zero
;                                 D3 is the timeout from the above trap #3
        trap #3                 ; Do it (No errors)
        tst.l d0                ; Did it work?
        bne sui                 ; No, exit via SUI

        subi.l #$C0008,(a1)     ; Subtract 12 from the width and 8 from the height
        lea wd0,a3              ; Get the address of window definition
        move.l #ww0_0,d1        ; Get the size of the working definition
        bsr getsp               ; Easy PEasy routine to ALCHP memory and set A0
        movea.l a0,a4           ; Which we save in A4
```

So far so good. Next we use a generic piece of code to go through the status area and set all the loose item status bytes to 'available'.

```
        lea wst0,a1             ; Status area starts here - thanks to SETW
        movea.l a1,a0           ; Copy to A0
        moveq #wst0_e-wst0-1,d1 ; How many bytes to clear - 1

st_clr  clr.b (a0)+             ; Clear one byte
        dbf d1,st_clr           ; Then the remainder
```

At this point we are just about ready to go. So, the next piece of code will call out the various WMAN routines to setup the window definition, position it on screen where the pointer currently is located and draw it before vanishing into the twilight zone that is the read pointer loop within WMAN. All of these have been described before so I don't go into detail.

```
        movea.l id(a6),a0       ; Get the channel id where we need it
;                                 A1 is the status area address from above
;                                 A3 is the window definition address from above
;                                 A4 is the working definition address from GETSP above
        move.l wd_xmin+wd_rbase(a3),d1 ; Get the minimum dimensions from definition
        andi.l #$0FFF0FFF,d1    ; Mask off any scaling factors - as previously described
        jsr wm_setup(a2)        ; Set up the working definition - no errors

        moveq #-1,d1            ; We want to draw the window where the pointer is
        jsr wm_prpos(a2)        ; So position it first as a primary window, then
        jsr wm_wdraw(a2)        ; Draw the contents
wrpt    jsr wm_rptr(a2)         ; Enter the "read pointer" loop in WMAN
```

At this point, WMAN takes over and we never get beyond the above code unless an event is detected – or set in an action routine – or an action routine flag an error. You will learn more about this as we add some meat to this program's workings later on.

The following code will exit from the program if an error occurred. The Z flag is already set or unset according to the value in D0.

```
        beq.s no_err            ; Since D0 is zero D4 is non zero
        bra sui                 ; An error occurred exit via SUI
```

If we are here, we need to check for any events that may have been detected or set in an action routine. In this example, we don't check every event, only the CANCEL event caused by the ESC key being pressed.

On return from the wm_rptr call, A0 is the channel id and A4 is the working definition address.

```
no_err  movea.l (a4),a1         ; Status area address
        btst #pt__can,wsp_weve(a1) ; Check for CANCEL event
        bne sui                 ; Exit

        bra.s wrpt              ; No more events, read pointer again
```

As mentioned above, this example currently is only checking for the ESC key being pressed. However, we have a loose item that can also be clicked to escape from the program. Rather than handling the ESC key and the loose item separately, we simply set the CANCEL event within the loose item action routine and let WMAN take care of it by passing control out of the read pointer loop into the above event handling code.

The following is the loose item action routine to do this.

```
; Loose item action routine

afun0_0 bset  #pt__can,wsp_weve(a1) ; Set the CANCEL event bit
        moveq #pt__can,d4         ; And load the CANCEL event number in D4
        moveq #0,d0               ; No errors
        rts                       ; Exit, and cause an exit from wm_rptr too
```

Because we have an application window defined, then the following is the default application window hit routine. When you hit the application window, or press the TAB key, the following code is executed. The default simply sets the registers to show no errors, no events and returns control back to the wm_rptr loop.

```
; Application sub-window hit routine

ahit0   moveq #0,d4
        moveq #0,d0
        rts
```

Note:

The loose item action routine and application window hit routine names, afun0_0 and ahit0 are hard coded by SETW and, unless we physically edit the code generated by SETW, we must use the names SETW chooses for us.

There is a pattern to the names though, ahit0 is the application sub-window hit routine for application sub-window zero. Ahit1 would be the hit routine for sub-window 1 and so on. For loose items, you have a layout number and a loose item number to contend with. So afun0_0 is for layout zero and loose item zero within that layout. Afun0_1 is the next loose item within that layout, and so on.

The remainder of the code, so far, consists of helper routines and is shown below without any further discussion.

```
; Various helper routines go here...

con     dc.w con_e-con-2          ; Size of channel definition
        dc.b 'con'
con_e   equ *

op_con  lea  con,a0               ; We want a console
        moveq #-1,d1              ; For this job
        moveq #0,d3               ; Timeout
        moveq #io_open,d0
        trap #2                   ; Do it
        rts
```

And finally, we need to load in the window definition we generated using SETW and all the Easy PEasy code libraries supplied by George.

```
; Pull in our window definition file.

        in  win1_source_ApplTestWin_asm

; We need George's Easy PEasy code next.

        in  win1_georgegwilt_peass_peas_sym_lst
        lib win1_georgegwilt_peass_peas_bin

; And finally, George's sprites.

        in  win1_georgegwilt_peass_csprc_sym_lst
        lib win1_georgegwilt_peass_csprc_bin
```

Save the code as "ApplTest_asm". At least, that's what I called mine!

The code above can be assembled and executed and the window we designed a couple of issues ago will be displayed on screen. Currently it does nothing useful but if you press the TAB key when the pointer is outside the application window (but inside the main window) then you will see it jump into the application window. HITting the 'X' loose item will cause the program to exit as will pressing the ESC key.

Next time, we will add some code to this example to allow us to monitor events. See you then.

Electronic QL Today

by Jochen Merz

Good News! The recent discussion ended in positive results: one of our trusty readers, Rainer Wolkwitz, sent me a sample scan in PDF, searchable and in excellent quality. He offered to scan the rest as well. So I sent him a large and very heavy box with all the QL Today masters (impossible to send them over the border, to Urs, for example) and he was very busy and scanned all of Volume 1 to 14 into PDFs. Again, in very good quality. The total size does not fit onto a CD (over 1 Gigabyte), so it needs to go onto a DVD.
We don't know what to do with it at the moment ... maybe burn it onto DVD and add it to the summer issue of this volume as a bonus. Or, wait until next volume and have all of the current volume 15 added to the DVD as well.

This is a very good way of turning something on paper into PDF - but it takes a few days and still cannot be done here by the editor, so no solution for immediate online delivery. Timing is often extremely tight, as I pick up the printed issue a day or two before they are to be shipped. But this is not the main problem: The files are way too large for email delivery anyway (one of the general problems we mentioned). We shall see if we find a solution to this in the future, but we feel for all the reasons discussed in detail that the delivery will remain on paper, at least for Volume 16.
So, all the historical work is preserved now for the future and, even better, it is searchable!
Thanks for the excellent work, Rainer - and thanks to all who offered to help as well!

Feedback

by Stephen Poole

This may well be the last article I write for quite some time. If readers wish, they may consult my Demo disk which should now be in the Quanta library. It contains quite a few programs which were too long to print in the magazines, and some which were. The first part is a copy of a microcartridge I sent to QL World demonstrating some methods of producing Vector Text when they said it was impossible on the QL, progressing to examples of 3D perspective text routines. QL World never bothered to reply, so, discouraged, I stopped writing programs for quite some time, until some ten years ago when I wrote a couple of letters to the Editors of Quanta and QL Today which ended up in print.. But most were in fact written between 1985 and 1995, and were rewritten and updated somewhat for publication as examples of simple programs, as I have seldom needed to write programs for my work since.

Other programs on the disk demonstrate various aspects of perspective animations with vectored text, and include a method of creating 3D animations of towns, countryside and scenery which was written long before NAVSATS used the same methods. But it must be said that most of these programs will only run completely on Original non-PC QL hardware, as they rely on peeking and poking which is not compatible with QPC. It seems odd that this old uncirculated output still cannot be achieved on QPC, so long live the QL! At present I have just run clean out of subjects to write about, and don't really have time to research new ones because of my present professional preoccupations, which concern inversing climate change simply and economically, which we have now developed but which we have to get adopted on a large scale. This takes much time lobbying.

Now on to Feedback : Computers will need much more time and development to match human or animal brains for one simple reason: Brains contain a colossal number of input channels and parallel and arborescent processing circuits. Information passes through innumerable chains of neurones, which possess hundreds of dendrites, which in turn connect to hundreds or thousands of other neurones in hundreds or thousands of processing centres. These dendrites cause the neurones they interact with to behave in different ways, reinforcing or weakening the different neurone's activity. The figures involved are so colossal that they are simply inconceivable to us. Yet all this happens within a sort of complex jellyfish weighing less than a kilogram, the whole lot coded onto about a meter-long strand of protein at birth. Anyone beat that for compression? This activity is feedback at all levels. OK, so supercomputers are massively parallel, but incorporating feedback at all levels is just physically impossible with current technology. Feedback has been studied by systems experts for a long time and forms the basis of simple SERVO SYSTEMS, used for example by tracking stations. Feedback is also known in electronics as that terrible harsh whistle you get when microphones form closed loops when badly situated in regard to loud speakers. Many writers complain that they never get any feedback from readers. I have generally found this to be the case, but no feedback is better than negative reactions. But feedback is vital in, for example, the scientific community or indeed to politicians, but most public opinion polls show that people seldom react. If Quanta and QL Today are to continue, they really do need more feedback from readers. I found it very stimulating when readers responded to the editor on certain subjects, and greatly enjoyed the exchanges, forcing me to react in consequence. So I am now looking for someone to replace me as a contributor. From conversations at shows I know that many people who never put pen to paper often have very interesting subjects to converse about. Believe me, once you start writing, it soon becomes a pleasure, especially as it forces you to sort your ideas out and helps you personally progress. So on behalf of your (extremely) hardworking Editor, please don't be shy, make a start and contribute something. I am sure we will all be delighted to read of your QL experiences. As a first subject, why not revise Jan Jones's excellent SuperBasic Handbook section on parsing diagrams where you will see how feedback is the basic logic of the language. If you are looking for a project, look at 'ELSE' and try to modify the parsing diagrams so that ELSE only feeds back as it should. Remember, democracy cannot survive without efficient feedback! So the magazines are in your hands....

The Next Volume

As you know from the past, we try to explain as much as possible to our readers - good news and bad news. We feel that if you understand the reasons, you feel better connected to the magazine and to us, the people "behind" QL Today.

We have not raised the price for some years, and the prices were based on an average issue of 32 pages. If we look back, we produced many more pages - every issue. This is very good news for you, the readers, and for us, as it shows we still have authors who write for us (one of the main worries, as articles often arrive very late and we also often fear we cannot fill an issue). Still, Steve's call on the other side of this page is true - no excuse for not writing!

Printing costs have gone up, and postage too.

Thanks to the help of Tony Firshman, the payment situation of the outstanding QL Today money from QBranch has been brought to a solution - by the time you read this, it should hopefully be settled. This has helped and put me into a slightly more relaxed situation about the QL Today costs.

As the number of subscribers got lower and lower, the costs per issue got higher and higher. Also, shipping to the UK readers from the UK stopped being feasible, as the difference in postage did not make up for the parcel containing all the UK issues. This was true for the last 4 or 6 issues, I can't remember exactly. This means, there has been nothing to save on the UK issues for some time - although we sold the subscription at a lower rate. The same is true for the Netherlands. I have not sent the issues from Holland for a while as the postage went up and up and up, and even the internal rate was raised at the beginning of this year.

Postage is one of the main factors, and as we take payment in advance for a period of a complete year, we don't know what is going to happen with the postage.

As the postage in Germany for letters inside Germany remained stable, we can keep the rate. As I will not be able to send all the issues from Austria in the future (travelling has become too expensive) I need to add 1 EUR to the subscription price for Europe, and treat all European readers the same as the costs are the same - this does not even cover inflation, but should hopefully pay for two issues sent from Germany and maybe two issues sent from Austria (likely).

Extremely good news for our readers outside of Europe: I can reduce the subscription price by 5 EUR! The reason: Deutsche Post has dropped the price for letters outside of Europe - you may remember that I complained for years that the postage was way too high! Maybe too many companies sent their letters from other countries, so they had to do something - I don't know.

Please renew as soon as possible to keep costs down - every reminder costs money and as you can see, I'm trying very hard to keep the price as stable as I can - but this needs your help too (and you've been helpful in the past), this is why I count you, the readers, as a stable factor!

Producing QL Today for you has been a big pleasure most of the time, and I hope it continues this way.

The Next Issue

You will probably remember that the three previous "Summer" issues were always early.

Although summer is usually the reduced-computing time for some of us, it had to do with the fact that I sent the issues from Austria in May (one month early). Last year, it was combined with the great QL meeting at Vienna. This year, no summer trip to Austria is planned, so it cannot be combined with a fixed, pre-planned trip.

Also, the DVD may need to be produced - maybe we can manage a bonus DVD with all the issues including all of volume 15. We shall see, no promises.

As summer time is usually holiday time too, and as the production of PDFs not only depends on me, I would ask you to be flexible this time to allow no fixed date for the schedule of issue 4 of this volume - just some time in June or July.

Please send your articles and submissions as soon as possible so that we are able to deliver another jam-packed bumper issue.

And please return the renewal form as soon as possible - we do depend on you in several ways, as you know.

So, let me end this issue with a sentence I often read myself:

QL forever!